Projekt „Netzwerk Wildkatze“ – Korridorstudie: Darstellungen von Barrieren und Vernetzungsnotwendigkeiten in Österreich

Josh Lowry and Luka Kern

Januar, 2024

natur schutz bund

Waldfonds Republik Österreich
Eine Initiative des Bundesministeriums für Land- und Forstwirtschaft, Regionen und Wasserwirtschaft

In Kooperation mit
Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie

Auftraggeber: Naturschutzbund, Museumplatz 2, A-5020, Österreich
Titelfoto: Daniel Leissing

Inhaltsverzeichnis

1 Einleitung

Die europäische Wildkatze (*Felis sylvestris*) wurde in Österreich in der Mitte des letzten Jahrhunderts bis zu ihrem Aussterben bejagt (Spitzenberger, 2005). In den letzten Jahren jedoch wurde die Präsenz mehrerer Wildkatzen Individuen in ganz Österreich, abgesehen von Salzburg, aufgenommen. Schließlich konnte bei Wildkatzen in Wachau, Niederösterreich, mithilfe von DNA Analysen auch belegt werden dass diese Nachkommen gezeugt hatten (Gerngross et al., 2021b). Obwohl die Wildkatze von der IUCN als nicht gefährdet (least concern) eingestuft wird, ist die Population in ihrem gesamten europäischen Verbreitungsgebiet generell rückläufig (Gerngross et al., 2021a). Zum heutigen Tag sind die Habitate der Widkatzen Populationen stark fragmentiert (Gil-Sánchez et al., 2020). Fragmentierte Habitate, in denen Populationen isoliert von anderen leben, sind anfälliger für genetische Engpässe und lokales Aussterben (Hansson et al., 1992). So haben sich beispielsweise die wachsenden Wildkatzen Populationen in der Schweiz sehr anfällig für die Introgression mit Hauskatzen erwiesen, wenn Individuen in Gebiete abgewandert sind, in denen es keine anderen Wildkatzen gibt (Nussberger et al., 2018; Quilodran et al., 2019). Aus diesen Gründen ist die Vernetzung der einzelnen Habitate für das Bestehen dieser Populationen wichtig; zudem könnte es sich als hilfreich erweisen Informationen darüber zu sammeln, wo eine Ausbreitung wahrscheinlich ist, um das Phänomen der Introgression zu überwachen.

Um dem Rückgang der Wildkatzen Population entgegenzuwirken hat der Naturschutzbund Österreich das Projekt "Rückkehr der Wildkatze" ins Leben gerufen. Als Teil dieses Projektes soll die vorliegende Arbeit eine Habitatanalyse und Darstellung der Vernetzung und Barrieren hinsichtlich der aktuellen Wiederbesiedlung der Wildkatze bieten. Um potentielle Wildkatzen Korridore zu identifizieren wurden simulierte Ausbreitungsdaten der Wildkatze, die mit dem prozessbasierten RangeShifter Modell generiert wurden, mit Landnutzungsdaten kombiniert und anschließend an einen Pfadfindungsalgorithmus übergeben. Der Umfang der Studie umfasst ganz Österreich sowie grenzüberschreitende Lebensräume, aus denen Individuen immigrieren. Die Ergebnisse können dazu genutzt werden um konkrete Korridor Planungen anzustoßen sowie um einen Überblick für die räumliche Anordnung der Habitate und deren Vernetzungsmöglichkeiten zu erhalten.

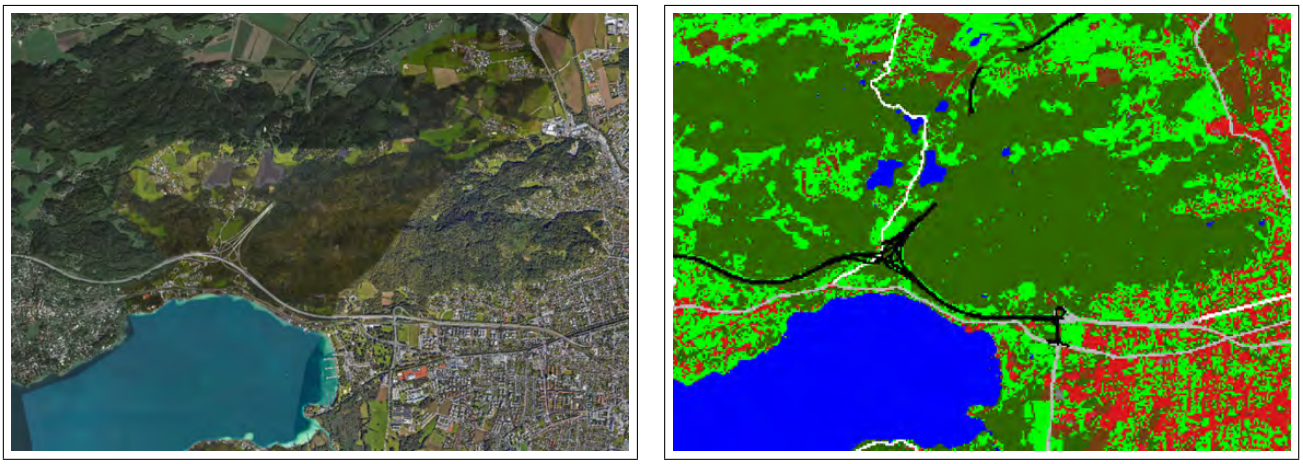
2 Methodik

Im Folgenden werden die zentralen Arbeitsschritte detailliert beschrieben: wie anfangs eine Landnutzungsanalyse zur Identifizierung von Habitaten basierend auf aktueller Literatur durchgeführt wurde. Wie diese Analyse, zusammen mit einem ausgewählten Parametersatz, genutzt wurde um Simulationen mit dem prozessbasierten Ausbreitungsmodell RangeShifter anzustoßen. Und wie die Simulationsergebnisse schließlich zusammen mit Landnutzungsdaten an einen Pfadfindungsalgorithmus übergeben wurden um potentielle Korridore zu identifizieren.

2.1 Identifizierung von Habitaten und Simulation der Ausbreitung

2.1.1 Landnutzungsdaten

Als Grundlage für die Habitatanalyse dienten Rohdaten zur Bodenbedeckung von Venter and Sydenham (2020), die auf der Webseite zenodo.org heruntergeladen und in die benötigte Auflösung von 20 m^2 überführt. Das Rohraster wurde dann anschließend jeweils einmal räumlich auf jedes Bundesland Österreichs zugeschnitten und dabei um 15 km gepuffert, sodass eine ausreichend große Fläche als Quellhabitate für das jeweilige Ausbreitungsgebiet mit berücksichtigt wurde.



(a) Satellitenbild (b) Landnutzungskarte

Abbildung 1: Beispiel der manuellen Überprüfung auf Tunnelabschnitte (schwaze Linie).

Neben natürlichen Barrieren wie Gewässern und hohen Gebirgszügen sind Straßen für die Ausbreitung der Wildkatze von großer Bedeutung, da dort viele Individuen getötet werden (Bastianelli et al., 2021; Westekemper et al., 2021). Aus diesem Grund wurde hier auch ein besonderes Augenmerk auf das Straßennetz Österreichs geworfen. Dazu wurden frei zugängliche Straßendaten von Geofabrik GmbH and OpenStreetMap Contributors (2023) verwendet. Mithilfe dieser Daten wurden drei Straßenklassen definiert, die unterschiedlich starke Auswirkungen auf die Ausbreitung von Wildkatzen haben sollen: Autobahnen und Fernstraßen wurden zu einer Autobahnklasse zusammengefasst, da es sich bei diesen um die Straßen mit den höchsten Geschwindigkeiten, dem höchsten Verkehrsaufkommen und der höchsten Anzahl von Fahrspuren handelt; diese Qualitätsmerkmale machen sie zu bedeutenden Hindernissen für Wildkatzen. Sogenannte Hauptverkehrsstraßen stellen Verbindungsstraßen zwischen größeren Städten dar. Sie werden mit geringeren Geschwindigkeiten befahren und sind kleiner als die Straßen der Autobahnklasse. Allerdings sind sie stärker befahren und immer noch als stark befahren einzustufen (Pinto et al., 2018). Sekundärstraßen wiederum verbinden oft kleinere Städte und werden dabei mit noch geringeren Geschwindigkeiten befahren und sind kleiner als die Hauptverkehrsstraßen. Straßenverläufe mit Tunneln wurden nachträglich manuell überprüft um die Tunnelabschnitte in den Landnutzungskarten nicht als Straße darzustellen (Abbildung 1), da diese sonst falsche

oberirdische Barrieren dargestellt hätten.

Die resultierende Landnutzungskarte (Abbildung 2) wurde anschließend einmal auf jedes Bundesland zugeschnitten um die nachfolgenden Arbeitsschritte effizient auf kleinere logische Einheiten anwenden zu können; eine Ausnahme bilden dabei Tirol und Vorarlberg sowie Kärnten und Osttirol, die jeweils zusammengefasst bearbeitet wurden.

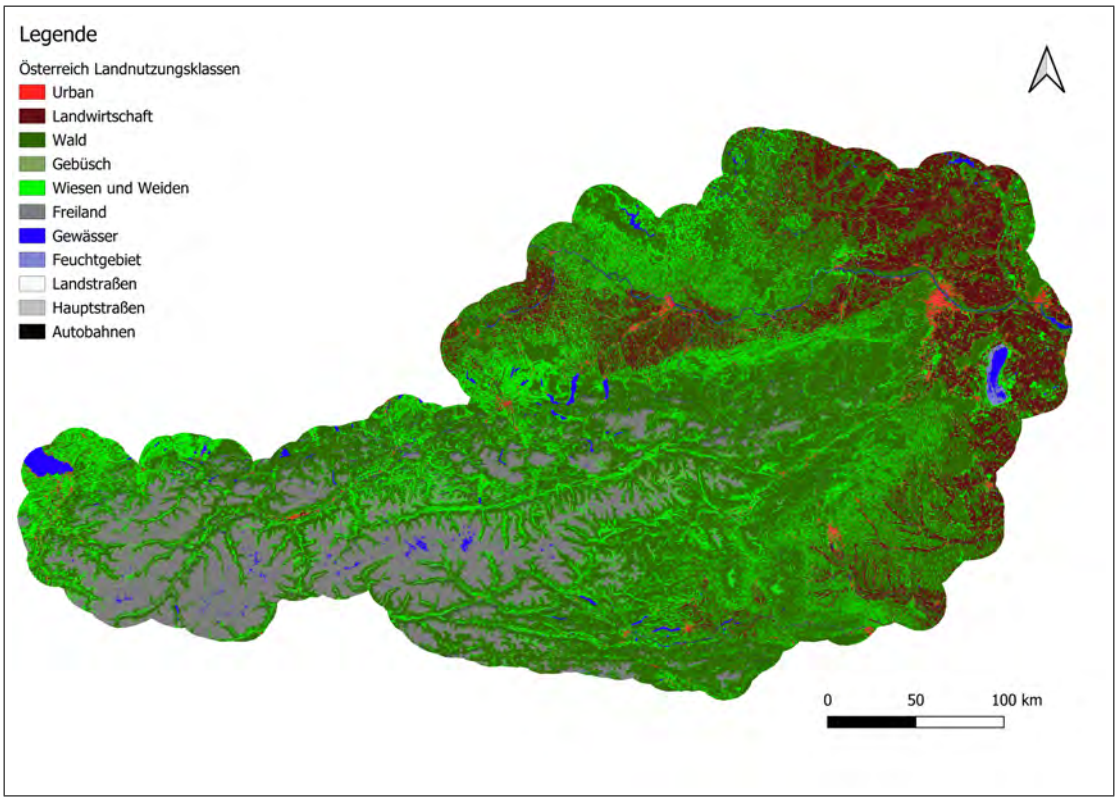


Abbildung 2: Landnutzungskarte Österreichs.

2.1.2 Analyse der Schneebedeckung

Eine hohe Schneedecke über einen ausgedehnten Zeitraum im Jahr erschwert die Überlebensbedingungen für Wildkatzen, da die Jagd auf Nagetiere schwieriger wird und alternative Strategien, wie das Fressen von Aas, angenommen werden müssen (Piechocki, 1990; Krofel et al., 2021). Informationen zur Schneebedeckung wurden daher verwendet, um die Gebiete aus den Untersuchungsflächen heraus zu nehmen, in denen an mehr als 100 Tagen im Jahr Schnee liegt (Slotta-Bachmayr et al., 2012). Die Daten zur Schneebedeckung stammen von der Euromammal-Datenbank [1]. Die Rohdaten beinhalteten Raster, in welchen die Zellwerte den Tagen mit Schneebedeckung entsprechen. Diese Daten waren für den Zeitraum von 2000 bis 2019 vorhanden; so konnte für diese Zeit ein Mittel berechnet werden. Das resultierende Raster wurde über die Landnutzungskarte gelegt, sodass die Zellen aus der Landnutzungskarte gelöscht werden konnten, in denen im Mittel an mehr als 100 Tagen im Jahr Schnee liegt. Das Ergebnis

[1] `https://euromammals.org/` Download am 12.02.2021)

war die Landnutzungskarte mit den Gebieten, in denen im Mittel an weniger als 100 Tagen im Jahr Schnee liegt (Abbildung 3).

Die gleiche räumliche Analyse der Schneedecke wurde zusätzlich einmal nur mit den Daten für 2019 wiederholt (Abbildung 4). Der Vergleich sollte als Beispiel dienen um zu zeigen wie drastisch sich die Winterbedingungen geändert haben. Die globalen Temperaturen sind in den letzten 20 Jahren gestiegen und werden in Zukunft aufgrund des Klimawandels voraussichtlich weiter ansteigen (Kikstra et al., 2022), wobei auch in den Alpen Vegetationsverschiebungen vorhergesagt werden (Lamprecht et al., 2018). Das bedeutet folglich, dass sich die Habitate aller Organismen verändern und verschieben werden, so auch für die Wildkatze.

Aus der Analyse der Schneebedeckung ergeben sich zwei Raster für die nächsten Arbeitsschritte, aus denen Habitate der Wildkatze extrahiert werden konnten: siehe auch Abbildung 3 und Abbildung 4.

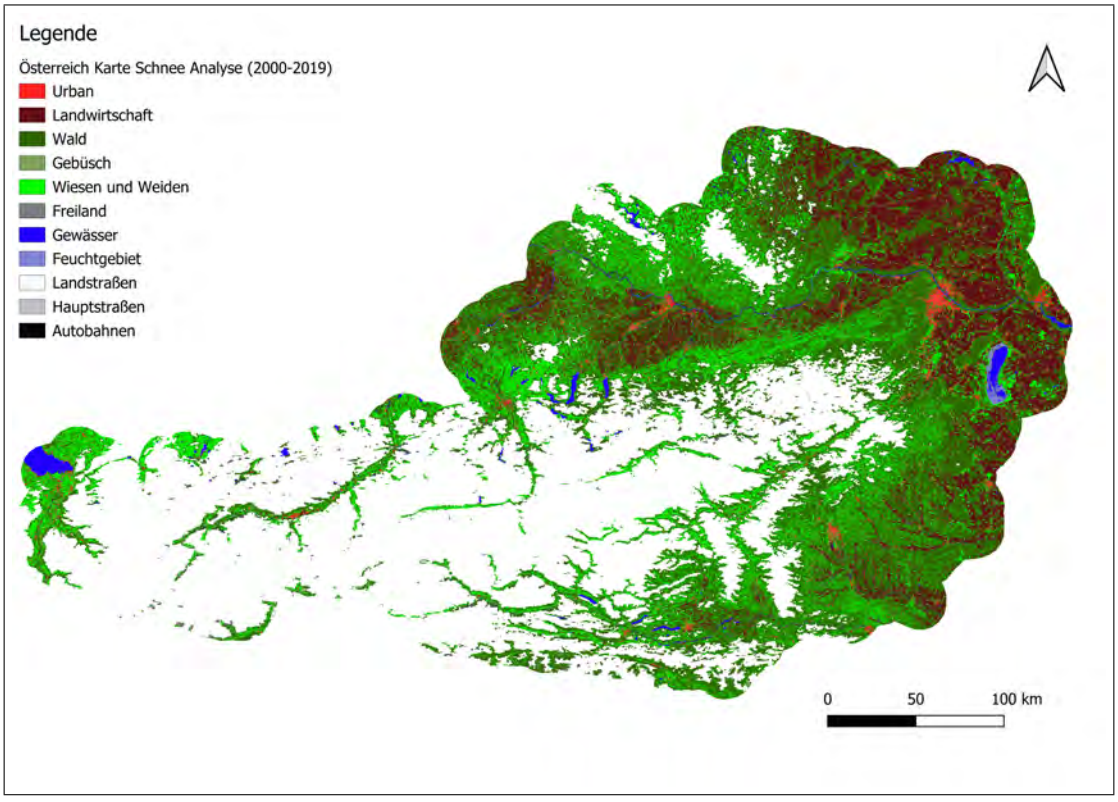


Abbildung 3: Landnutzungskarte ohne Gebiete, in denen die Schneebedeckung in Tagen pro Jahr für den Zeitraum 2000-2019 im Mittel mehr als 100 Tage im Jahr betrug.

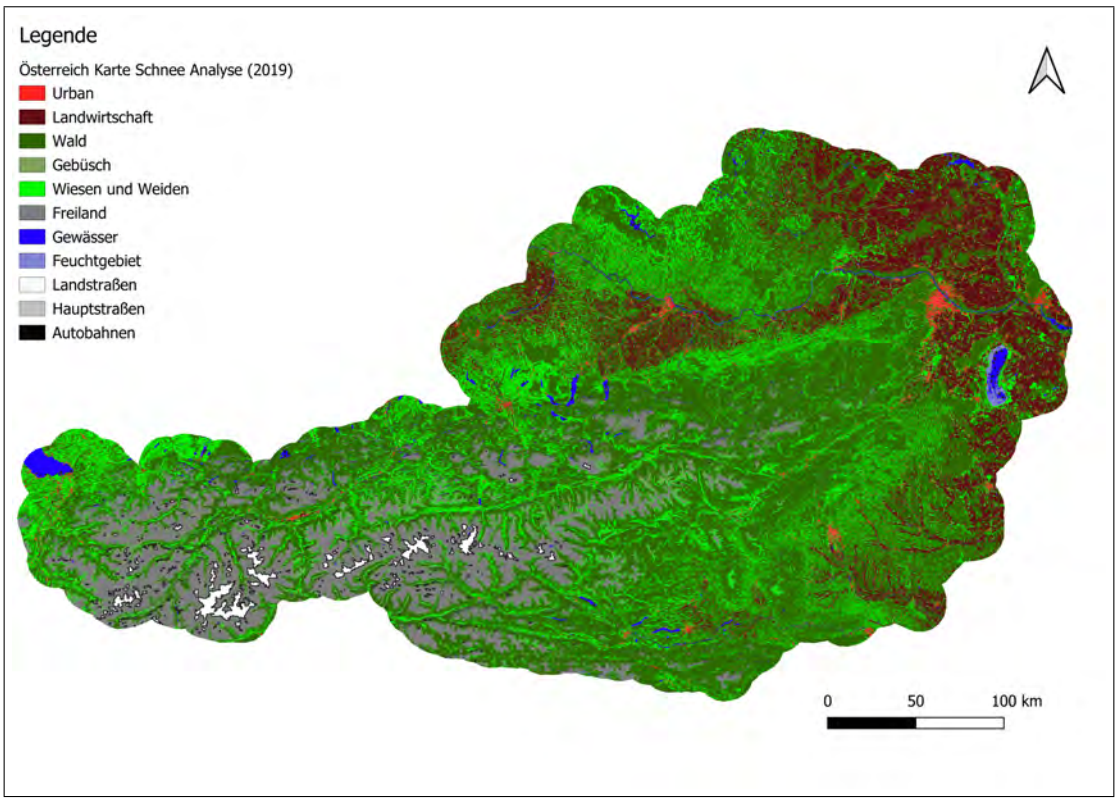
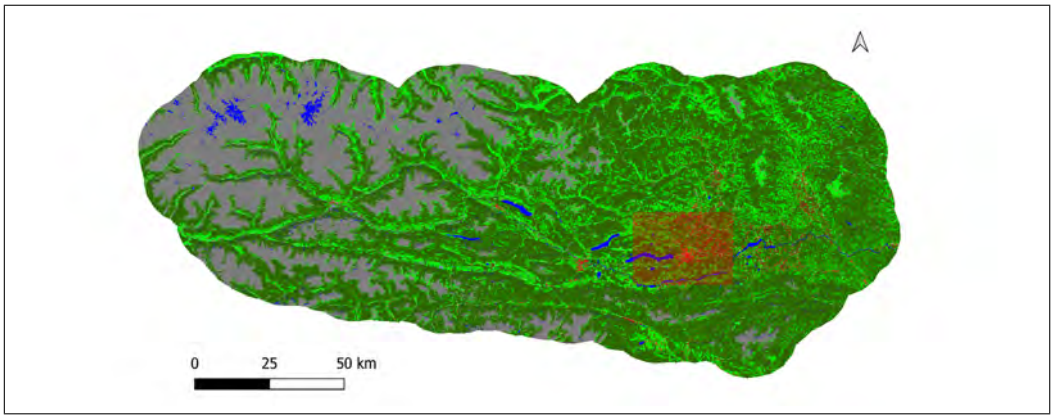


Abbildung 4: Landnutzungskarte ohne Gebiete, in denen die Schneebedeckung in Tagen pro Jahr im Jahr 2019 mehr als 100 Tage im Jahr betrug.

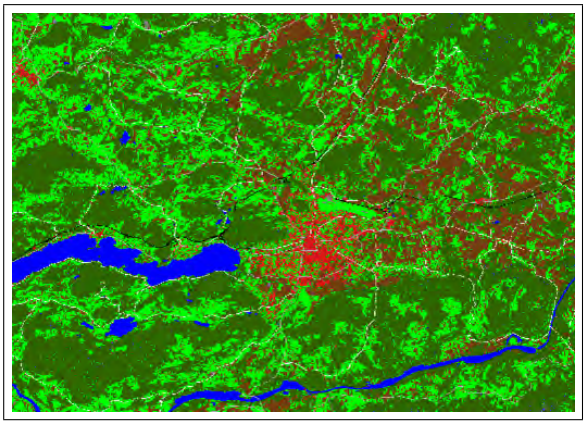
2.1.3 Identifizierung von Habitaten

Auf Grundlage der Landnutzungskarten konnten im Zusammenspiel mit einer Literaturrecherche Wildkatzen Habitate bestimmt werden. Im folgenden werden die Anhaltspunkte aus der Literatur aufgeführt. So haben frühere Studien über Wildkatzen eine Lebensraumpräferenz für Laubwälder festgestellt (Friembichler and Slotta-Bachmayr, 2013). Neuere Studien jedoch zeigen, dass Wildkatzen durchaus auch andere Lebensräume wie Nadelwälder (Bourdouxhe et al., 2020) und Agrarlandschaften (Jerosch et al., 2017) nutzen. Basierend darauf wurden in dieser Arbeit alle Waldklassen zu einer einzigen Klasse zusammengefasst. Flächen mit einer Größe von weniger als 100 ha wurden entfernt. Die verbleibenden Waldflächen wurden um 50 m gepuffert. Grünland, Gebüsch und landwirtschaftliche Flächen innerhalb des 50 m Puffers wurden beibehalten, da sie wichtige Beutequellen für die Wildkatze darstellen. Allerdings hat sich gezeigt, dass die landwirtschaftliche Intensität die Größe des Verbreitungsgebiets der Wildkatze beeinflusst, was jedoch in der vorliegenden Arbeit nicht berücksichtigt werden konnte. Streif et al. (2017) konnten zudem zeigen dass Wildkatzenweibchen die Distanz von 50 m nur selten überschreiten wenn sie aus der Dickung herauskommen. Städtische Gebiete, kahles Land, Autobahnen, Hauptverkehrsstraßen, große Flüsse, Gewässer und Feuchtgebiete wurden entfernt, da sie nicht zu den Lebensräumen der Wildkatze zählen. Sekundärstraßen wiederum befinden sich nachweislich innerhalb der Heimatgebiete der Wildkatze (Bastianelli et al., 2021) und wurden daher innerhalb des Puffers belassen, um eine Trennung von Lebensräumen zu vermeiden.

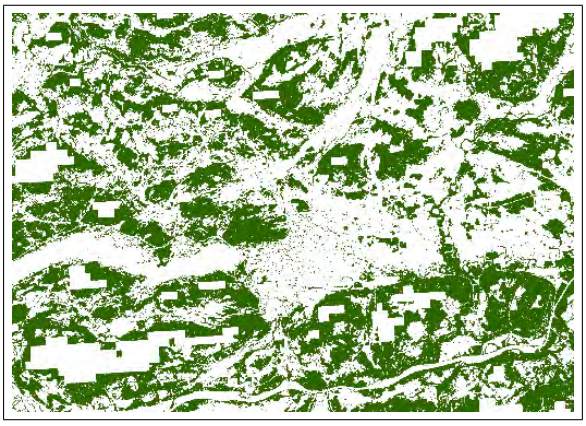
Der so generierte Habitat-Layer wurde zusätzlich so gefiltert, dass alle Flecken mit einer Gesamtfläche von weniger als 1 500 ha entfernt wurden, um die durchschnittliche Größe des Verbreitungsgebiets entsprechend Bastianelli et al. (2021) zu berücksichtigen (Abbildung 5). Auf diese Weise wurden ausschließlich Flächen berücksichtigt, die mindestens ein Wildkatzen Paar beherbergen konnten. Anschließend wurden aus den definierten Habitaten Flächenstatistiken extrahiert, um die Auswahl der Korridore weiter zu unterstützen.



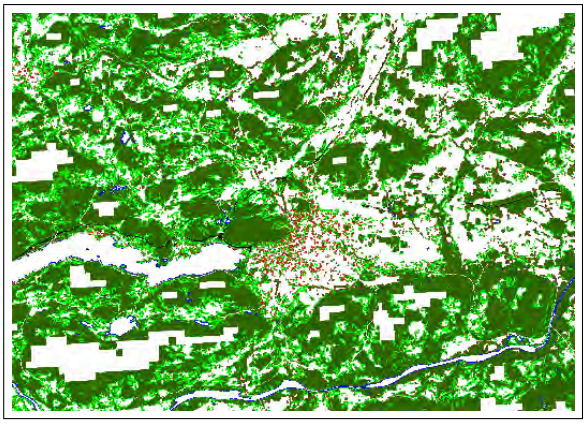
(a) Landnutzungskarte Klagenfurt.



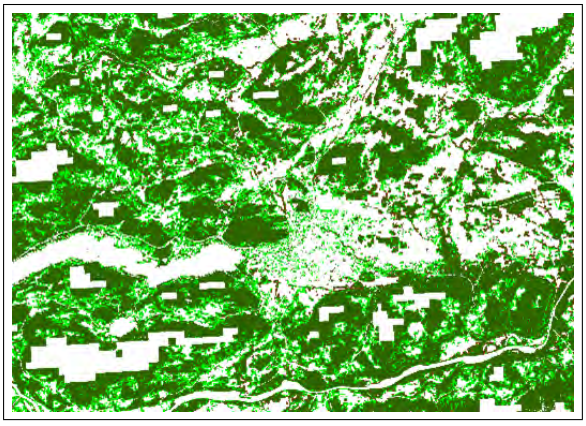
(b) Zoom roter Bereich s. (a).



(c) Extraktion von Waldflächen.



(d) 50 m Puffer um Waldflächen.



(e) Entfernung von ungeeigneten Flächen.

Abbildung 5: Erstellen von Habitat Layern. Legend der Landnutzungsklassen: Urban (rot), Landwirtschaft (braun), Wald (dunkelgrün), Gebüsch (graugrün), Wiesen und Weiden (hellgrün), Freiland (hellgrau), Gewässer (blau), Feuchtgebiet (hellblau), Landstraßen (weißgrau), Hauptstraßen (dunkelgrau), Autobahnen (schwarz).

2.1.4 Modellkalibrierung und Simulation

Die Dynamiken der Wildkatzenpopulation wurden mit dem R-Paket RangeShiftR (Malchow et al., 2021) simuliert, das auf der Grundlage der Modellierungssoftware RangeShifter V2 (Bocedi et al., 2021) funktioniert. RangeshifterR ist ein pozess- und individuenbasiertes, räumlich explizites Modell, welches das Ausbreitungsverhalten (Bocedi et al., 2014) in Kombination mit demografischen Informationen simuliert und es somit ermöglicht, die Ausdehnung des Verbreitungsgebiets (Dominguez Almela et al., 2020) sowie Wiederansiedlungsszenarien (Ovenden et al., 2019) zu modellieren.

Um eine Ausbreitungssimulation durchführen zu können, benötigt RangeshifterR demografische Daten wie Auswanderungs-, Transfer- und Siedlungswerte. Die in dieser Arbeit verwendeten Parameterwerte stammen aus Lowry et al. 2023 QQQ (s. Anhang 5.1), mit Ausnahme der Überlebensraten für Jungtiere (Tabelle 1) und der Werte für die Transferkosten sowie die Mortalität (Tabelle 2).

Die Transferkosten beschreiben die Wahrscheinlichkeit, mit der ein Individuum in eine Rasterzelle einer bestimmten Landnutzungsklasse zieht, im Verhältnis zu der Wahrscheinlichkeit, mit der es in eine Rasterzelle mit geeigneten Lebensraum zieht. In dem vorliegenden Fall entspricht der geeignete Lebensraum der Landnutzungsklasse "Wald", welche den Referenzwert 1 erhält. Für die landwirtschaftliche Landnutzungsklasse standen zwei Werte aus Bourdouxhe et al. (2020) zur Auswahl, 80 oder 500. Wir haben für die Simulationen dieser Arbeit den höheren Wert als konservativer Schätzung gewählt, da es für Österreich nicht bekannt ist ob und in welchem Ausmaß die Wildkatze landwirtschaftliche Flächen betritt. Derselbe Transferkosten Wert wurde auch für Grasland, Shrubland und urbane Gebiete gewählt. Alle anderen Landnutzungsklasse habe den Wert 1000 bekommen, da sie mit einer noch geringeren Wahrscheinlichkeit von Wildkatzen betreten werden.

Die Mortalitätskosten sind die Wahrscheinlichkeit, dass ein Individuum stirbt, wenn es in eine Rasterzelle einer bestimmten Landnutzungsklasse zieht. Straßen sind bekanntlich die häufigste Todesursache bei Wildkatzen (Klar et al., 2009, 2012), und können als Barrieren zwischen Populationen den genetischen Austausch behindern (Hartmann et al., 2013; Würstlin et al., 2016; Westekemper et al., 2021). Ziel war es die Auswirkungen der durch Straßen verursachten Mortalität als Barriere bestmöglich abzubilden. So wurde die Mortalität durch Straßen mit den Methoden von Kramer-Schadt et al. (2004) ermittelt. Bastianelli et al. (2021) errechneten, dass 54 % der Wildkatzen durch Straßenkollisionen starben. Um die Mortalität durch Straßen zu definieren wurde folglich eine Gesamt-Mortalität und eine Baseline-Mortalität bestimmt. Als Gesamt-Mortalität wurde die 75 % Sterblichkeitsrate von Jungtieren gewählt, ermittelt von Lange and Götz (2014). Als Baseline-Mortalität wurden 37 % festgelegt, sodass die Baseline-Mortalität nicht zu gering ist und die Differenz zwischen Baseline-Mortalität und Gesamt-Mortalität in der Größenordung für Straßen-Mortalität von Bastianelli et al. (2021) liegt. Basierend auf dieser Differenz wurden schließlich die Sterblichkeitsraten auf der Straße

schrittweise angepasst, so dass die Sterblichkeitsrate insgesamt wieder bei etwa 75 % lag.

Tabelle 1: Für die Simulationen verwendete demographische Lesley-Matrix. Geburtenraten in der obersten Zeile und Überlebensraten als Prozentsatz von 1 in den Zeilen darunter.

	Jungtier männlich	Jungtier weiblich	Erstes Jahr männlich	Erstes Jahr weiblich	Zweites Jahr männlich	Zweites Jahr weiblich
Jungtier	0	0	0	2.9	3.7	3.7
Erstes Jahr (m) männlich	0.63	0	0	0	0	0
Erstes Jahr (w) weiblich	0	0.63	0	0	0	0
Zweites Jahr (m) männlich	0	0	0.79	0	0.85	0
Zweites Jahr (w) weiblich	0	0	0	0.9	0	0.93

Tabelle 2: Die Transferkosten Werte der unterschiedlichen Landklassen, die für den stochastischen Bewegungssimulator verwendet wurden. Die Werte wurden mit der datenbasierten Methode von Bourdouxhe et al. (2020) ermittelt.

Landnutzungsklassen Kodierung	Landnutzungsklassen	Transferkosten Wert	Sterblichkeits- pro Bewegung
1	Urban	1000	0.008
2	Landwirtschaft	500	0.0001
3	Wald	1	0.0001
4	Gebüsch	500	0.0001
5	Wiesen und Weiden	500	0.0001
6	Freiland	1000	0.0001
7	Gewässer	1000	0.0001
8	Feuchtgebiet	1000	0.0001
9	Landstraßen	1000	0.0009
10	Hauptstraßen	1000	0.003
11	Autobahnen	1000	0.15

Mit den definierten Parametersätzen konnten anhand der IUCN-Wildkatzenverbreitungskarte von Gerngross et al. (2021a) besetzte Gebiete ausgewählt, von denen aus die Ausbreitung am ehesten beginnen würde (s. die gelben Gebiete in Abbildung 33). Diese Ausgangslage wurde an RangeshifteR übergeben und die Ausbreitung simuliert; die Anzahl der Wildkatzen, die dabei in den jeweiligen Habitatstücken starten, basierte auf den aktuell in der Schweiz gefundenen Individuen Dichten (Maronde et al., 2020). Die Ausbreitung der Wildkatze wurde dann 100 mal für jedes Bundesland simuliert, wobei jede Simulation einem Zeitraum von 20 Jahren abbildete. Für die auf den Simulationen aufbauenden Arbeitsschritte wurde dann aus den 100 Wiederholungen das Mittel errechnet.

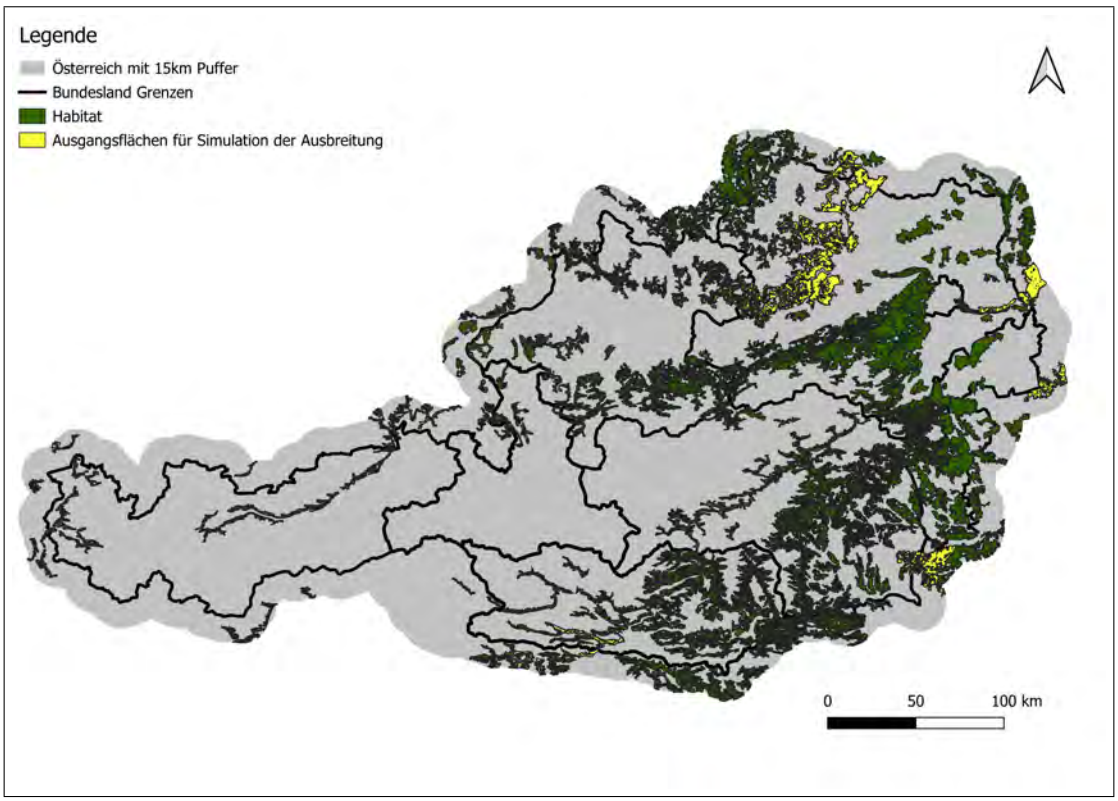


Abbildung 6: Habitat Flächen, die aus der Habitatanalyse hervorgehen. Die gelb hervorgehobenen Gebiete stellen Lebensräume dar, die basierdend auf Gerngross et al. (2021a) als Ausgangsflächen für die Ausbreitungssimulation ausgewählt wurden.

2.2 Potentielle Korridore automatisiert identifizieren

2.2.1 Ziel der Methode

Das Ziel dieses Arbeitsschrittes war es automatisiert potentielle Korridore für zu detektieren. Die Grundlage bieten großflächig vorhandene Informationen über Wildkatzen Habitate, Bewegungsmuster von Wildkatzen und die Landnutzung durch den Menschen, die in den vorangegangenen Arbeitsschritten herausgearbeitet worden sind. So sollen benachbarte Wildkatzen Habitate auf möglichst kurzem Weg verbunden werden, wobei die simulierten Bewegungsmuster und die Landnutzungsklassifikation berücksichtigt werden müssen. Drei Skripte bieten einen Lösungsansatz für diese Zielstellung. Im Folgenden werden die Eingangsdaten und die Vorgehensweise in den Skripten beschrieben. Die Skripte befinden sich zudem im Anhang dieses Berichtes (s. Appendix 5.3).

2.2.2 Eingangsdaten

Die Eingangsdaten sind raumbezogene Daten für verschiedene Bundesländer in Österreich: Kärnten, Steiermark, Burgenland, Oberösterreich, Tirol, Vorarlberg, Niederösterreich und Salzburg. Die verwendeten Eingangsdaten umfassen:

- `Patches.shp`: Eine Shape Datei mit Polygonen, welche die beobachteten Habitatzonen von Wildkatzen beschreibt. Jedes Polygon repräsentiert ein Habitatgebiet.
- `Heatmap.tif`: Eine Rasterdatei, die die gemittelten Ergebnisse von Modellrechnungen enthält und potentielle Wildkatzenbewegungsmuster darstellt. Die Zellenwerte sind dabei die Besuchshäufigkeit (Anzahl).
- `Landuse.tif`: Eine Rasterdatei, die eine Klassifikation der Landnutzung mit entsprechenden Kodierungen für verschiedene Landnutzungsformen enthält.

2.2.3 Skripte

Detektieren von potentiellen Korridoren

Das Skript `PotentialCorridorFinder.py` ermöglicht das systematische Finden potentieller Wildkatzenkorridore zwischen benachbarten Habitaten in den verschiedenen österreichischen Bundesländern. Ziel dieses Skripts ist es, Bewegungsmuster von Wildtieren wie Wildkatzen sowie Landnutzungsformen zu berücksichtigen um potentielle Korridore zu identifizieren. Dabei nutzt das Skript räumliche Datenverarbeitungstechniken und arbeitet in folgenden Schritten:

1. **Identifikation von Nachbarhabitaten:** Das Skript beginnt mit der Identifizierung von benachbarten Habitat Polygonen, die zwar räumlich benachbart sind, sich jedoch nicht überschneiden. Diese werden folgend als solche Habitat Polygone betrachtet, zwischen denen potentielle Korridore gesucht werden könnten.
2. **Eckpunkte identifizieren:** Für jedes Paar von benachbarten Habitat Polygonen werden die nächstgelegenen Eckpunkte auf deren Rändern bestimmt. Diese Eckpunkte fungieren als Start- und Endpunkte für die Korridore und werden als Paare gelistet. Von diesen Start- und Endpunkt Paaren werden dann möglichst viele Paare genutzt — höchstens jedoch zwanzig — um zwischen ihnen potentielle Korridore zu identifizieren.
3. **Erstellen eines Gewichtungsgraphen:** Weiterführend wird für die jeweils betrachtete Fläche (immer für zwei benachbarte Polygone) ein ungerichteter Graph erstellt, in dem für jede Zelle von `Heatmap.tif` und `Landuse.tif` ein Knoten generiert wird. Die Kanten zwischen den Knoten repräsentieren dabei deren Verbindungen. Diese Kanten werden gewichtet, wobei die Gewichtungen auf den Werten aus den Heatmap- und Landnutzungsdaten basieren. Bereiche mit höheren Werten in der Heatmap (die potentielle Lebensräume für Wildkatzen anzeigen) und vorteilhafter Landnutzung erhalten niedrigere Gewichtungen, was auf bevorzugte Verbindungen zwischen den Knoten hinweist.
4. **Pfadfindung und Identifikation von Korridoren:** Unter Verwendung des erstellten Gewichtungsgraphen werden potentielle Korridore zwischen den Eckpunkten ermittelt.

Die A*-Suche wird als Pfadfindungsalgorithmus verwendet, um die Pfade mit den geringsten Gesamtgewichtungen zu finden. Diese Pfade repräsentieren mögliche Bewegungswege für Wildtiere zwischen den Habitaten.

Datenbearbeitung

Das Skript `LanduseProcessing.py` ist als Hilfsskript zu verstehen und bereitet die Daten für weiterführende Analysen vor. Es konvertiert die Rasterzellen anhand der spezifischen Landnutzungskodierung für Wald in binäre Rasterzellen und dann in Wald-Polygone. Diese Polygone zeigen direkt die räumliche Ausdehnung der Waldgebiete und können für weitere räumliche Bearbeitungsschritte genutzt werden. Während das Skript die Daten nicht direkt für die Analyse der Wildtierkorridore verwendet, legt es den Grundstein für die Korridorflächenberechnung, indem es die relevanten Informationen in einem geeigneten Format zusammenstellt.

Korridor Flächenanalyse

In dem Skript `CorridorAreaCalculation.py` werden die potentiellen Wildkatzenkorridore aus dem Skript `PotentialCorridorFinder.py` analysiert. Insgesamt führt dieses Skript die eigentliche Erstellung der Korridore durch und ermittelt dabei die Flächen der bewaldeten und der unbewaldeten Gebiete, die in den Korridoren liegen. Hier sind die Schritte des Skriptes aufgelistet:

1. **Puffern der Korridorlinien:** Die Korridorlinien werden um 50 m gepuffert, um einen Bereich um die Korridorlinien zu schaffen, der bei der Analyse berücksichtigt wird. Die gepufferten Linien werden in einen GeoDataFrame umgewandelt, der als Shape Datei speicherbar ist.

2. **Zuschneiden der gepufferten Linien:** Die gepufferten Linien werden anhand der Habitat Polygone zugeschnitten. Dadurch werden die tatsächlichen Flächen der potentiellen Korridore identifiziert.

3. **Identifizierung von Waldgebieten:** Durch Überlagerung der Wald Polygone aus `LanduseProces` und der potentiellen Korridore wird festgestellt, welche Anteile der Korridore bewaldet sind und welche nicht.

4. **Berechnung und Anzeige von Flächen:** Das Skript berechnet die Flächen der verschiedenen Bereiche; die gesamte Korridorfläche, die bewaldete und die unbewaldeten Fläche.

5. **Schreiben der Flächenstatistiken:** Die berechneten Flächen werden in eine Textdatei geschrieben, um die Ergebnisse zu dokumentieren.

6. **Speichern der kombinierten Korridorflächen:** Schließlich werden die Korridorflächen als Shape Datei gespeichert.

2.2.4 Selektion geeigneter Korridore

Es bleibt unbedingt zu erwähnen, dass die automatisierten Analyseschritte nur Informationen über potentielle Korridorflächen liefern. Die Flächen benötigen definitiv eine Validierung und Plausibilitätsprüfung durch Expert:innen. Das bedeutet, dass der Vorteil der Automatisierung vor allem darin liegt einen Überblick zu erlangen, gerade wenn es sich um große Gebiete handelt, die untersucht werden sollen. In einer auf persönlicher Expertise – durch Josh Lowry – basierenden Analyse, wurden anschließend an die automatisierte Erkennung für jedes Bundesland eine handvoll potentieller Korridore für die Umsetzung vorgeschlagen. Um dabei einen ersten Anhaltspunkt für die Auswahl zu bekommen wurde ein weiteres Mal die Besiedlungswahrscheinlichkeit genutzt, die mithilfe von RangeShifter simuliert wurde. Diese Information wurde mit den Informationen über aktuelle Wildkatzenvorkommen und dem Vorhandensein von Barrieren durch Straßen verbunden um final fundierte Vorschläge für potentielle Wildkatzenkorridore machen zu können.

3 Ergebnisse

3.1 Habitat Analyse

Die Tabelle 3 präsentiert zusammenfassend die Analyse potentieller Wildkatzen Habitate für die Bundesländer Österreichs (s. Abschnitte 2.1.1 und 2.1.2). Hierbei sind Unterschiede in der Anzahl potentieller Lebensräume, der größten Lebensraumfläche und der Gesamtlebensraumfläche für Wildkatzen aufgeführt.

So zeigt Salzburg 32 potentielle Lebensräume mit einer Gesamtlebensraumfläche von 65 835 ha, jedoch ohne einen tatsächlichen Nachweis für das Vorkommen von Wildkatzen; die potentielle Lebensraumfläche ist dabei die kleinste unter den Bundesländern. Im Gegensatz dazu weist Niederösterreich die mit Abstand größte Anzahl von Nachweisen für Wildkatzen auf, wobei auch die Gesamtlebensraumfläche mit 912 683 ha die größte unter den Bundesländern ist. Kärnten zeigt die zweit meisten Nachweise für Wildkatzen, während weder die Anzahl potentieller Lebensräume noch die gesamte Lebensraumfläche sich deutlich von den anderen abheben. Steiermark und insbesondere Oberösterreich heben sich in der Anzahl potentieller Lebensräume deutlich von den anderen Bundesländern ab; doch obwohl Oberösterreich nochmal 50 potentielle Lebensräume mehr verzeichnet als die Steiermark, weist die Steiermark fast mit 635 555 ha das doppelte an gesamter Lebensraumfläche auf, was sich auch in der größten zusammenhängenden Lebensraumfläche zeigt. Wenn nun noch das Burgenland und Tirol/Vorarlberg miteinander verglichen werden, so gibt es für beide Gebiete fast gleich viele Nachweise für Wildkatzen trotz einer vielfach größeren Gesamtlebensraumfläche des Burgenlandes.

Tabelle 3: Analyse der Habitat Daten für Bundesländer Österreichs; Schneebedeckung gemittelt über den Zeitraum 2000 bis 2019.

Bundesland	Anzahl potentieller Lebensräume	Größte Lebensraumfläche (ha)	Gesamte Lebensraumfläche (ha)	Anzahl von Nachweisen für Wildkatzen
Salzburg	32	6 789	65 835	0
Kärnten	54	29 799	404 638	33
Steiermark	92	47 083	635 555	16
Burgenland	47	53 928	330 604	9
Tirol/Vorarlberg	26	11 283	82 339	10
Niederösterreich	54	406 730	912 683	433
Oberösterreich	142	20 436	350 121	16

Die Tabelle 4 konzentriert sich auf die Ergebnisse der Habitatanalyse, bei der für die Schneebedeckung ausschließlich die Daten aus dem Jahr 2019 genutzt wurden. Auch hier sind die Unterschiede in der Anzahl potentieller Lebensräume, der größten Lebensraumfläche und der Gesamtlebensraumfläche für Wildkatzen aufgeführt.

Die Bundesländer Burgenland und Niederösterreich sind nicht aufgeführt, da die Schneebedeckung hier keinen Unterschied zu dem Mittel der Jahre 2000 bis 2019 gezeigt hat. Für die restlichen Bundesländer zeigen sich hingegen deutliche Unterschiede. In Salzburg hat sich die Anzahl potentieller Lebensräume fast verdoppelt, während sich sowohl die größte als auch die gesamte Lebensraumfläche ungefähr verzehnfacht hat. Für Kärnten zeigt sich in der Anzahl der potentiellen Lebensräume zwar kaum Veränderung, aber die größte als auch die gesamte Lebensraumfläche hat sich auch hier vervielfacht, ungefähr mit den Faktor drei. In der Steiermark ist sogar, wie auch i Oberösterreich, ein Rückgang in der Anzahl potentieller Lebensräume zu verzeichnen, die größte als auch die gesamte Lebensraumfläche ist aber auch hier auch deutlich größer. Für Tirol/Vorarlsberg ist ein deutlicher Anstieg in allen drei Kenngrößen zu sehen; besonders deutlich ist, dass die gesamte Lebensraumfläche hier um ein zehnfaches größer ist.

Tabelle 4: Analyse der Habitat Daten für Bundesländer Österreichs; Schneebedeckung aus dem Jahr 2019. Burgenland und Niederösterreich sind nicht aufgeführt, da die Schneebedeckung hier keinen Unterschied zu dem Mittel der Jahre 2000 bis 2019 gezeigt hat.

Bundesland	Anzahl potentieller Lebensräume	Größte Lebensraumfläche (ha)	Gesamte Lebensraumfläche (ha)
Salzburg	59	64 992	701 920
Kärnten	58	128 438	1 184 056
Steiermark	82	161 645	1 730 685
Tirol/Vorarlberg	78	76 929	871 679
Oberösterreich	72	98 795	880 003

Die Tabelle 5 gibt Aufschluss über die Barrieren durch Straßen, die potenzielle Wildkatzenkorridore durchschneiden. Hierbei wurde die Gesamtzahl der Barrieren durch Hauptstraßen

und Autobahnen für jedes Bundesland berücksichtigt. Besonders hervorzuheben ist Niederösterreich, das mit 882 Barrieren durch Hauptstraßen und 194 durch Autobahnen eine Gesamtzahl von 1 076 Barrieren aufweist. Im Gegensatz dazu verzeichnet Burgenland lediglich 142 Hauptstraßenbarrieren und 52 Autobahnbarrieren, was zu einer Gesamtzahl von 194 Barrieren führt. Diese Ergebnisse verdeutlichen die regionalen Unterschiede in der Fragmentierung von Wildkatzenlebensräumen durch Straßeninfrastruktur.

Tabelle 5: Barrieren durch Straßen, die potentielle Korridore durchschneiden.

Bundesland	Barrieren Hauptstraßen	Barrieren Autobahnen	Gesamtzahl der Barrieren
Salzburg	52	47	99
Kärnten	412	112	524
Steiermark	443	126	569
Burgenland	142	52	194
Tirol/Vorarlberg	131	36	167
Niederösterreich	882	194	1 076
Oberösterreich	547	97	644

3.2 Korridor- und Konnektivitätsanalyse

3.2.1 Burgenland

Die Simulation für die Ausbreitung der Wildkatze ergibt für die Habitate des Burgenlandes im Durchschnitt eine Besiedlungswahrscheinlichkeit von 12 %. Deutlich zu erkennen ist, dass insbesondere Gebiete an der Bundeslandgrenze im Süden und Norden höhere Besiedlungswahrscheinlichkeiten aufweisen; hier sind auch zwei Wildkatzennachweise verzeichnet. Gerade für diese Gebiete würden sich Korridore anbieten um die Ausbreitung der Wildkatze zu fördern. So liegen die vorgeschlagenen Korridore im Süden des Burgenlandes bei Poppendorf, Weichselbaum und Gerresdorf bei Güssing (s. Abbildungen 7 bis 9). Für das erste Gebiet sind drei Korridore zu empfehlen (s. Abbildung 8). Die Kennummern der Habitatflächen, auf die Bezug genommen wird, sind auch die Habitatgrößen in Hektar.Die ersten beiden befinden sich bei Weichselbaum und verbinden das Habitat mit der Kennnummer `53928` mit dem Habitat `1944`. Der dritte Korridor verbindet bei Poppendorf das Habitat `1944` mit dem Habitat `2439`. Das Habitat `1944` ist dabei wie ein Trittstein zwischen den beiden größeren Habitaten; problematische Stelle könnten allerdings die Straßenbarrieren bei Poppendorf sein. Das zweite Gebiet befindet sich bei Gerresdorf bei Güssing. Hier werden drei Korridore vorgeschlagen, die das Habitat `2439` mit dem Habitat `2240` verbinden sollen (s. Abbildung 9). Hier sind keine Barrieren vorhanden. Bei Betrachtung der Übersicht in Abbildung 7 lässt sich erkennen, wie die Etablierung dieser Korridore die Ausbreitung der Wildkatze im Süden des Burgenlandes begünstigen könnte.

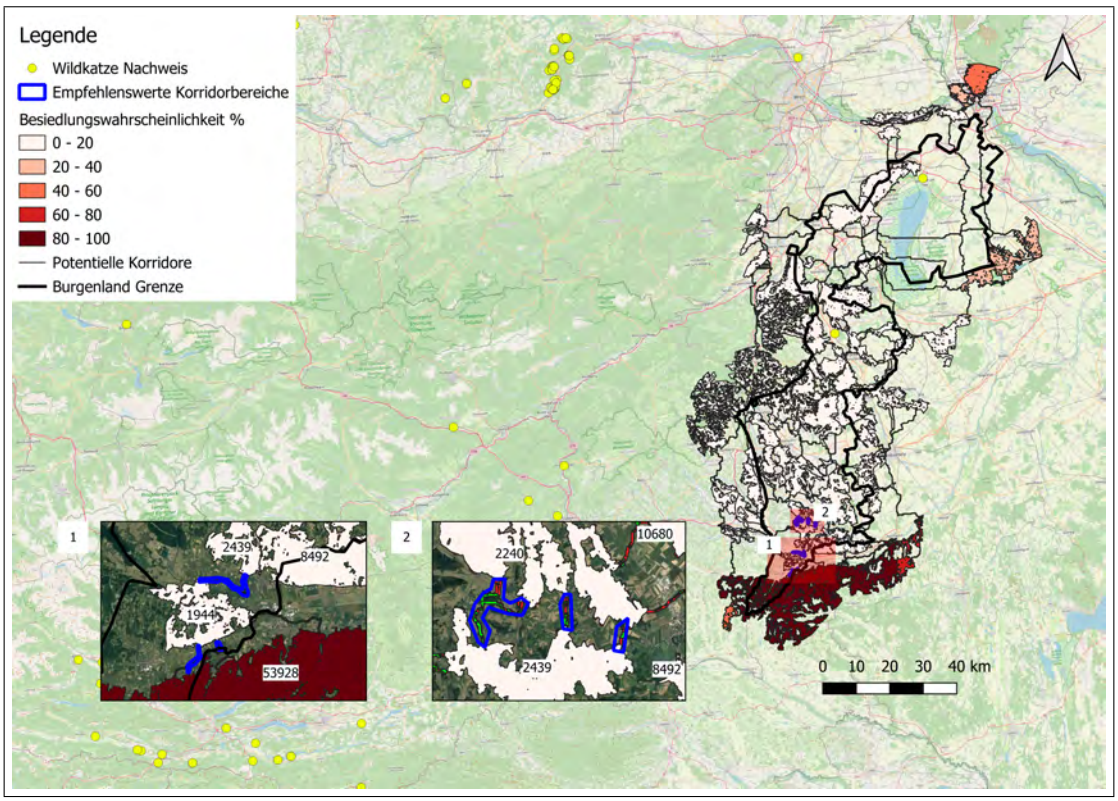


Abbildung 7: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen im Burgenland.

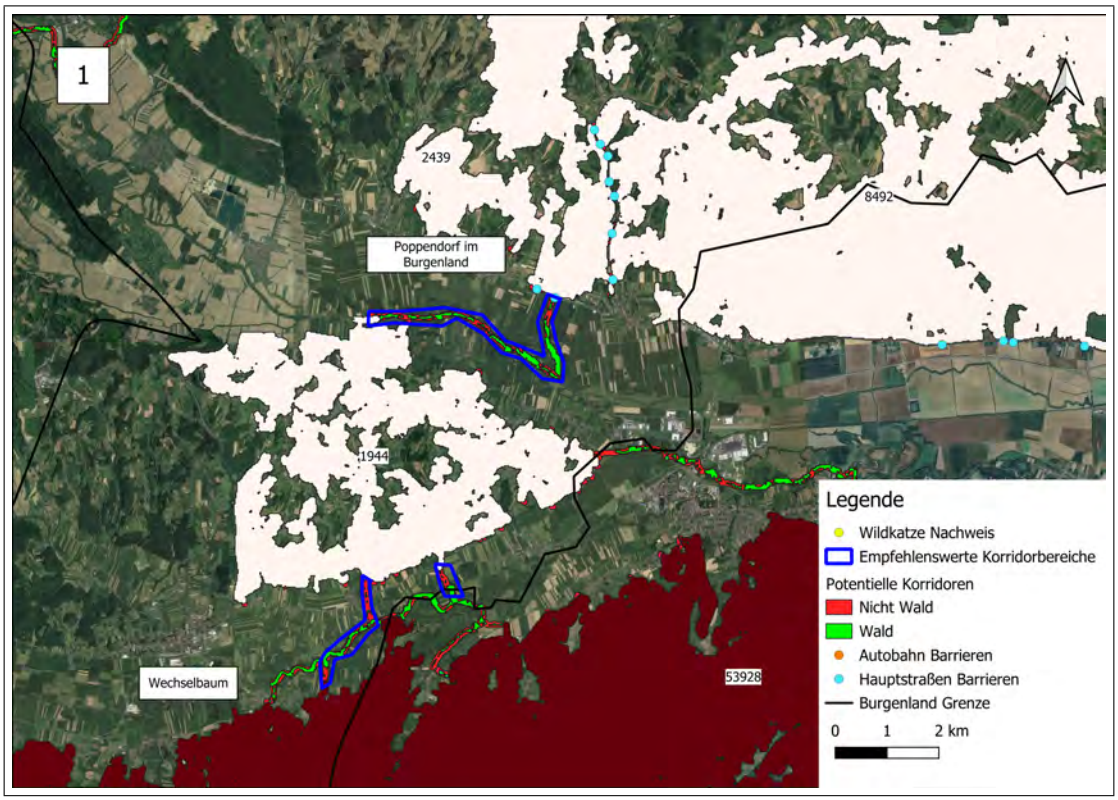


Abbildung 8: Erstes empfohlenes Korridorgebiet im Burgenland (s. Abbildung 7).

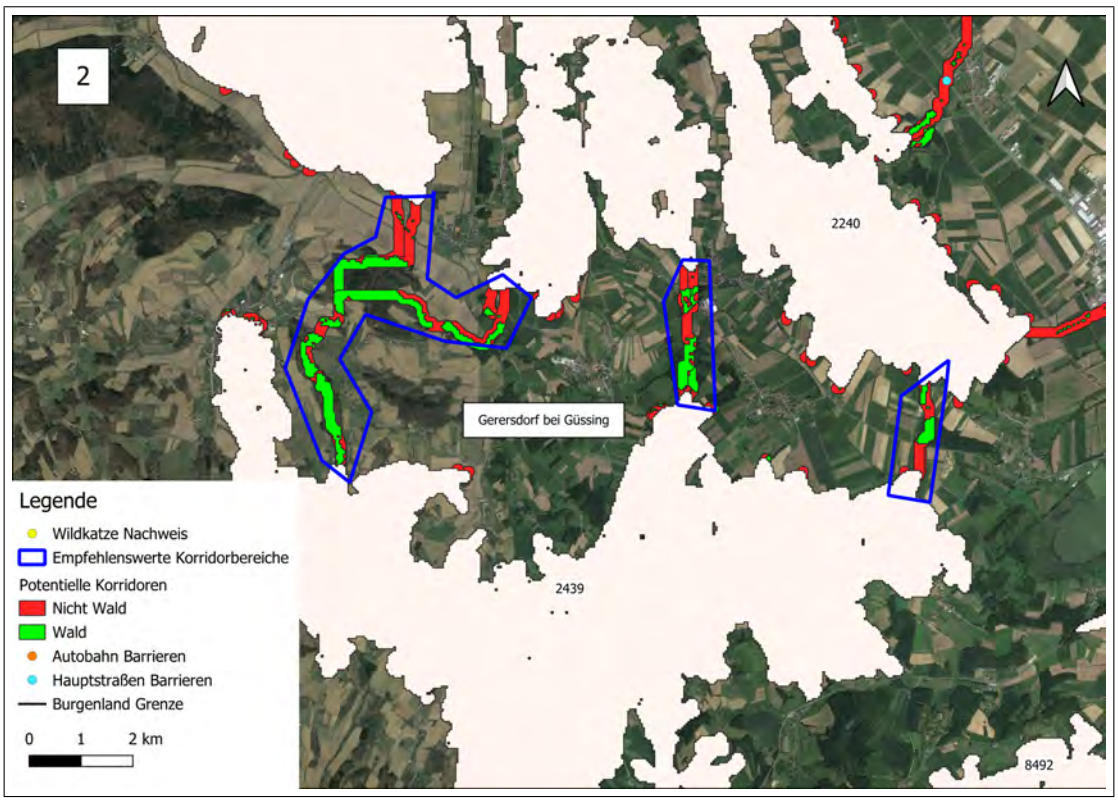


Abbildung 9: Zweites empfohlenes Korridorgebiet im Burgenland (s. Abbildung 7).

3.2.2 Steiermark

Für Habitate der Steiermark zeigt die Simulation der Ausbreitung im Mittel eine Besiedlungswahrscheinlichkeit von 27 %. Die Habitate mit den höchsten Besiedlungswahrscheinlichkeiten befinden sich an den Grenzen der Steiermark im Süden, Süd-Osten und Nord-Osten, während es aber auch Wildkatzennachweise im Zentrum der Steiermark gibt. Die vorgeschlagenen Korridorgebiete befinden sich im Süden und Süd-Osten und könnten potentiell die Lücke zwischen zwei größeren Gebieten mit höherer Besiedlungswahrscheinlichkeit schließen (s. Abbildung 10). Das erste Gebiet liegt im Süd-Osten bei Sankt Anna am Aigen und Kronnersdorf (s. Abbildung 11). Hier sollen die Habitate `31925`, `1786` und `1955` miteinander verbunden werden. Gerade der Korridor zwischen Habitat`1786` und Habitat `1955` wäre eine längere Überbrückung, bei der allerdings auch kleine Waldstücke al kleinste Trittsteine genutzt werden könnten. Ein Problem könnte allerdings die Barriere durch eine Hauptstraße darstellen. Die anderen beiden Gebiete liegen weiter im Westen. Das erste davon liegt bei Schamberg und Wieseldorf (s. Abbildung 12). Dabei sollen die Habitate `21702`, `2051` und `3874` verbunden werden. Auch hier können möglicherweise kleinere Waldstücke bei der Konstruktion von Korridoren von Hilfe sein. Allerdings ist auch hier die mögliche Problematik mit Barieren durch Hauptstraßen zu beachten. In dem anderen Gebiet befinden sich zwei Habitate (`21702` und `6648`) mit höheren Besiedlungswahrscheinlichkeiten (s. Abbildung 13); diese sind allerdings durch eine Autobahn getrennt. Ziel der vorgeschlagen Korridore in diesem Gebiet soll vielmehr sein diese beiden Habitate mit jeweils einem anderen Habitat mit niedrigerer Besiedlungswahrscheinlichkeit zu verbinden. So könnte das Habitat `21702` mit dem Habitat `3847` bei Pichling bei Stainz verbunden werden. Dabei wäre die Strecke recht kurz und es wäre auf der Fläche schon überwiegend Wald vorhanden, allerdings besteht auch eine Barriere durch eine Hauptstraße. Zudem könnte das Habitat `6648` mit dem Habitat `39608` bei Kleinsöding verbunden werden. Hier wäre die Herausforderung eine Barriere durch die Autobahn, aber möglicherweise lässt sich mit lokaler Expertise eine Lösung finden.

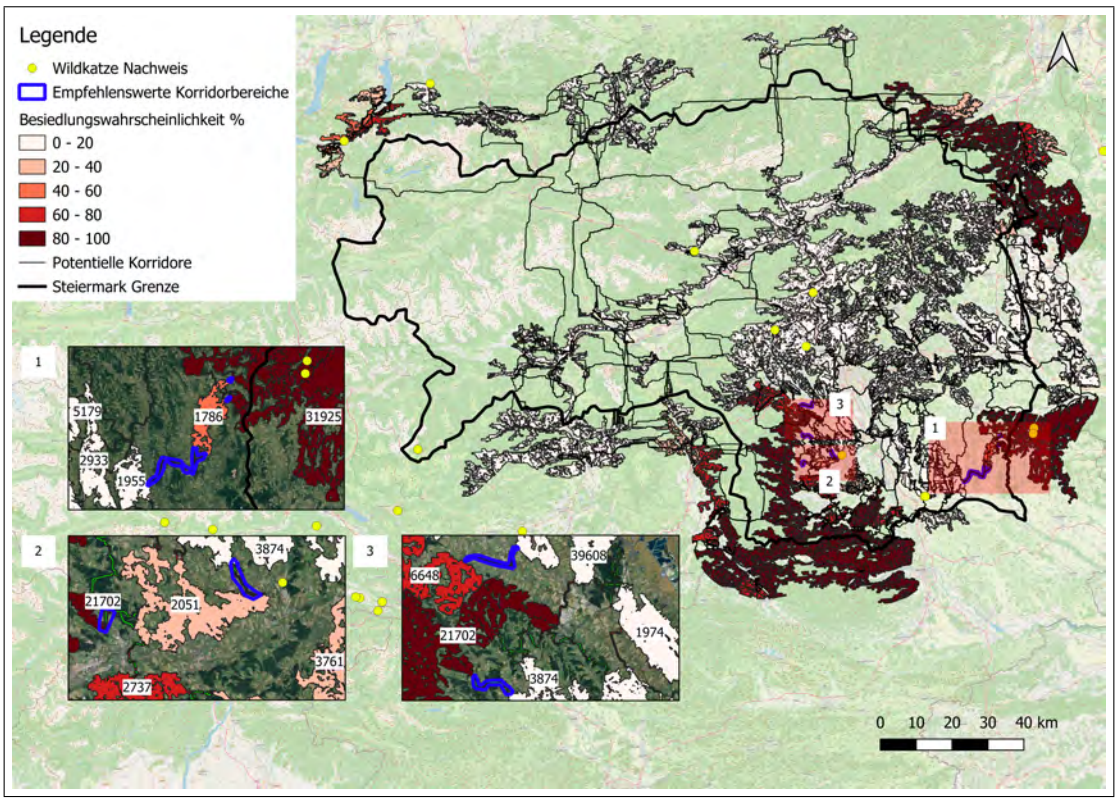


Abbildung 10: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in der Steiermark.

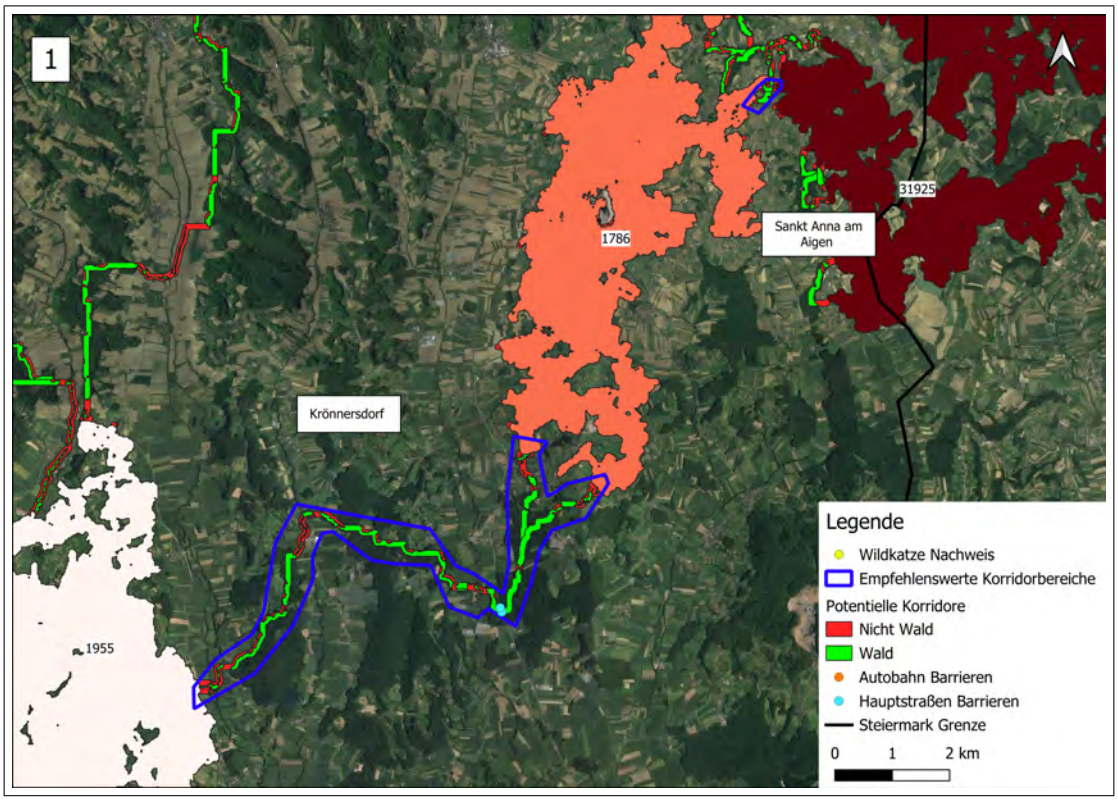


Abbildung 11: Erstes empfohlenes Korridorgebiet in der Steiermark (s. Abbildung 10).

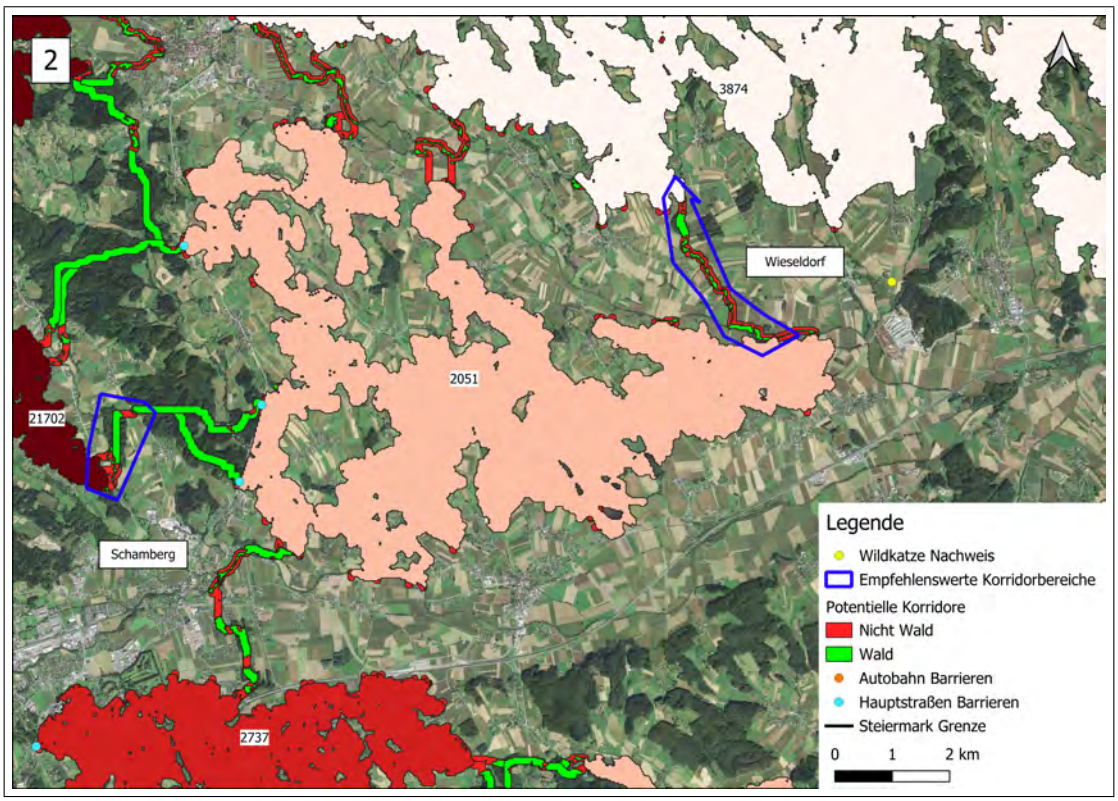


Abbildung 12: Zweites empfohlenes Korridorgebiet in der Steiermark (s. Abbildung 10).

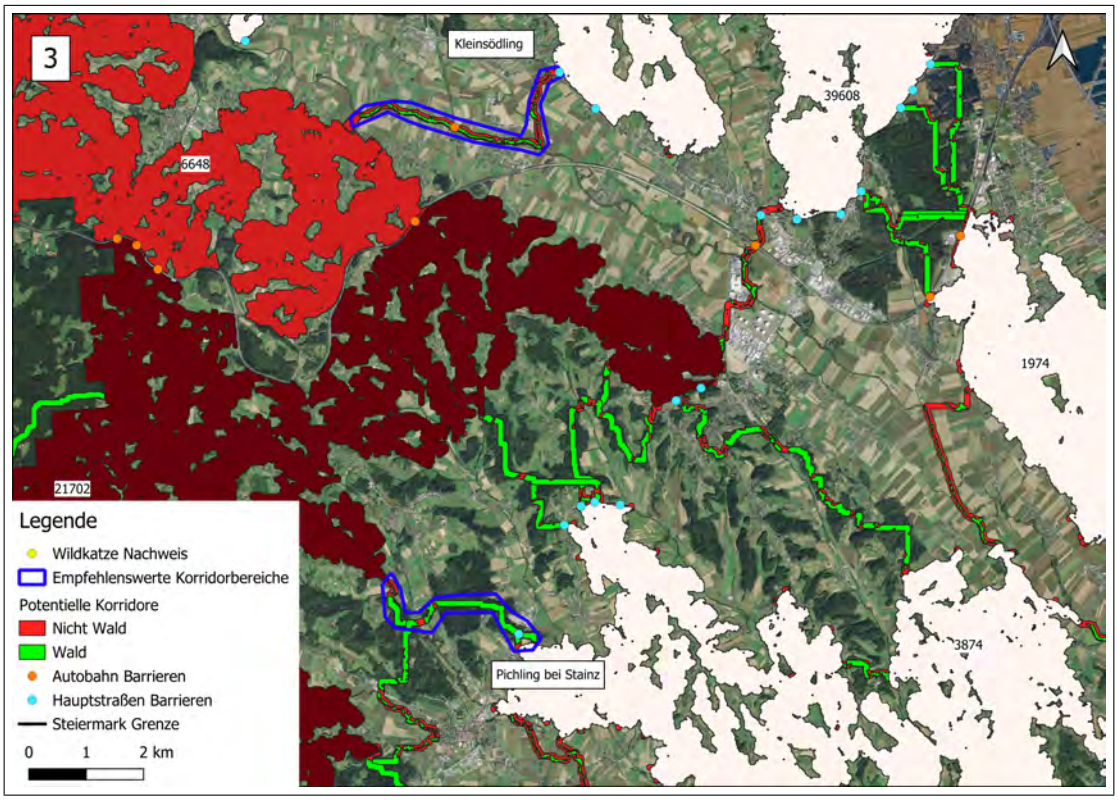


Abbildung 13: Drittes empfohlenes Korridorgebiet in der Steiermark (s. Abbildung 10).

3.2.3 Kärnten

Die Habitate in Kärnten zeigen in der Simulation mit Ausnahme des Norden eine recht hohe Besiedlungswahrscheinlichkeit; diese liegt im Durchschnitt bei 41 %. Die Nachweise für Wildkatzen sind verstreut über den Westen und Süden Kärntens. Die Gebiete mit vorgeschlagenen Korridoren befinden sich zentral in Kärnten und im Osten an der Bundesland Grenze (s. Abbildung 14). Das erste Gebiet liegt zentral bei Velden (s. Abbildung 15). Hier sind mehrere Möglichkeiten für Korridore vorgeschlagen. Allerdings ist das Gebiet von Hauptstraßen und Autobahnen geprägt. Alle möglichen Korridore kreuzen mindestens eine Hauptstraße oder Autobahn. Die Verbindung zum Habitat `7519` wäre allerdings wertvoll, da es einen Weg in die Habitate mit niedrigeren Besiedlungswahrscheinlichkeiten darstellen würde. Im zweiten Gebiet bei Stein im Jauntal weiter östlich sind an einer Stelle zwei mögliche Korridore vorgeschlagen. Diese würden das Habitat `5165` mit dem Habitat `2912` verbinden (s. Abbildung 16). Allerdings besteht auch hier die Herausforderung mit Barrieren durch Hauptstraßen und Autobahn. Im dritten Gebiet noch weiter östlich bei Bleiberg besteht an zwei Stellen die Möglichkeit das Habitat `3844` mit jeweils einem Habitat mit höherer Besiedlungswahrscheinlichkeit zu verbinden. Auch hier bestehen Barrieren durch Hauptstraßen, jedoch keine durch Autobahnen (s. Abbildung 17).

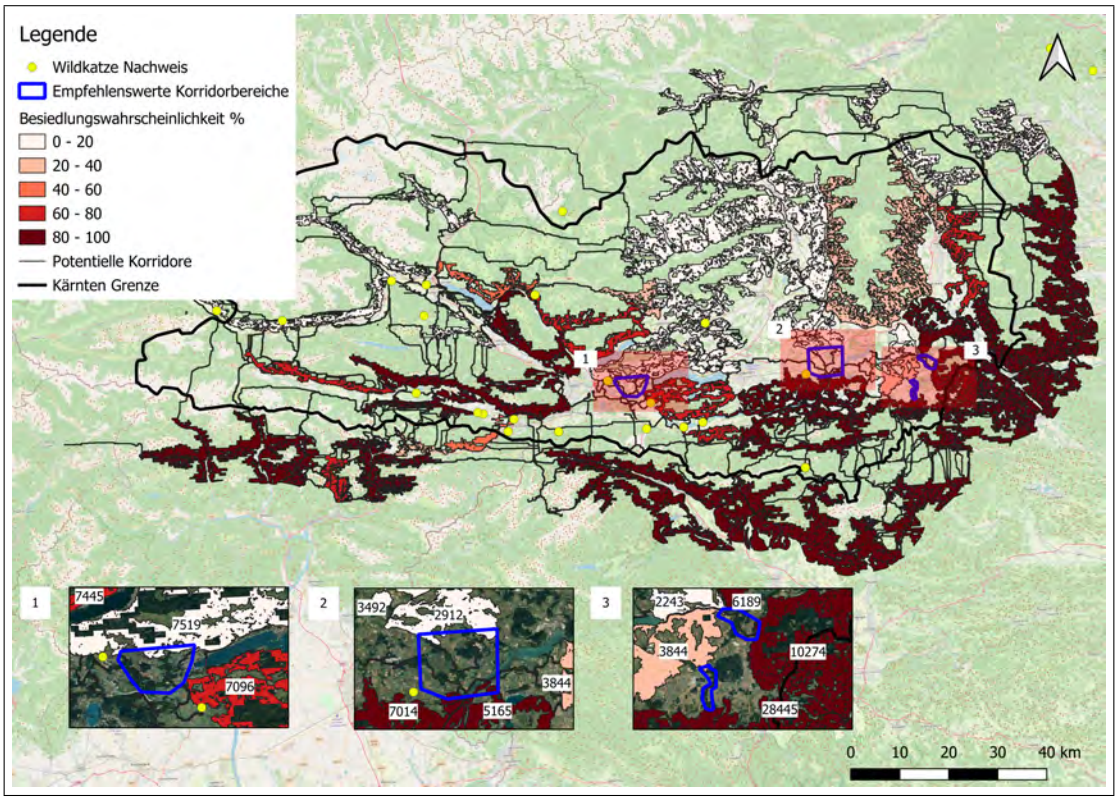


Abbildung 14: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in Kärnten.

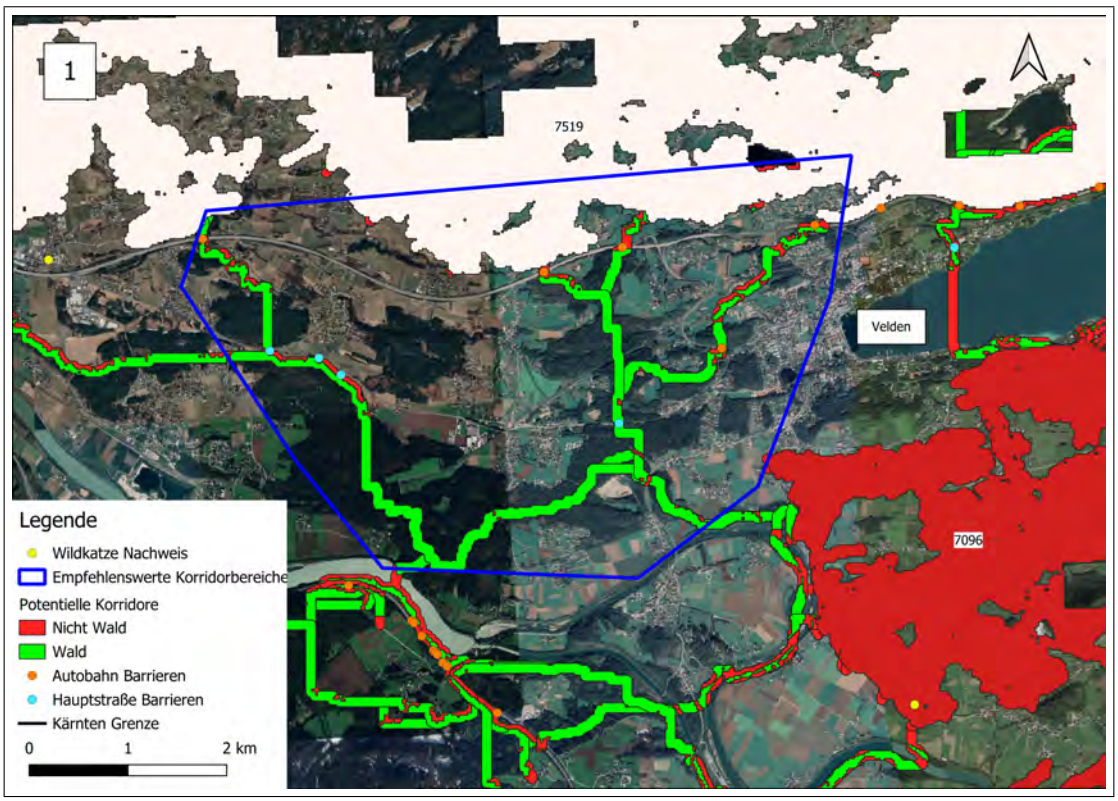


Abbildung 15: Erstes empfohlenes Korridorgebiet in Kärnten (s. Abbildung 14).

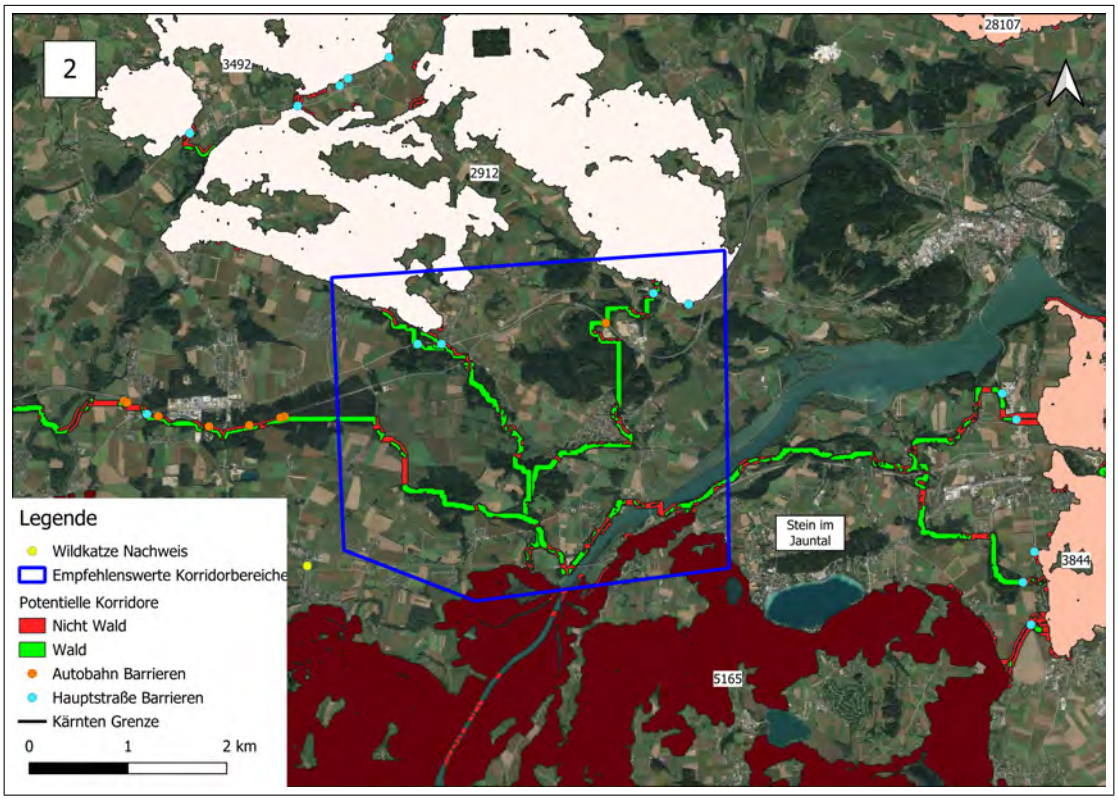


Abbildung 16: Zweites empfohlenes Korridorgebiet in Kärnten (s. Abbildung 14).

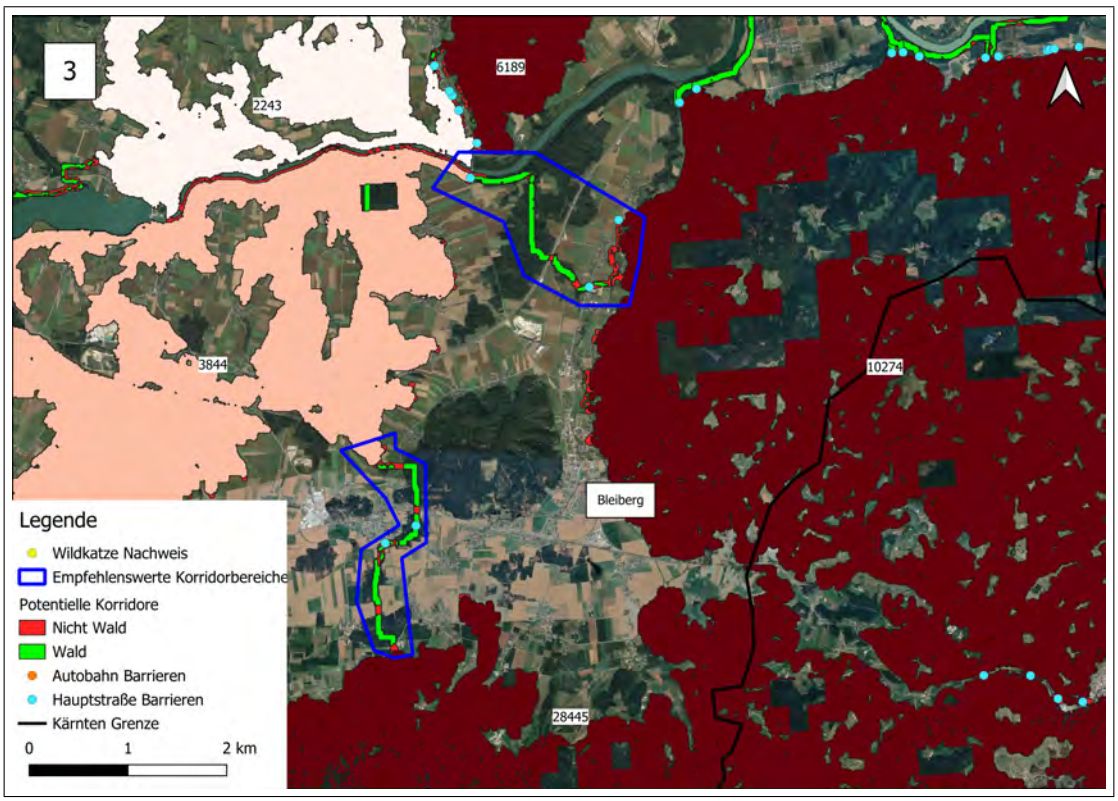


Abbildung 17: Drittes empfohlenes Korridorgebiet in Kärnten (s. Abbildung 14).

3.2.4 Niederösterreich

Die Habitate in Niederösterreich, die in der Simulation eine höhere Besiedlungswahrscheinlichkeit zeigen, liegen vor allem im Nord-Westen oder auch an der Grenze im Nord-Osten. Die Habitate im Nord-Osten zeigen zudem einige Nachweise für Wildkatzen. Die ersten beiden Gebiete, für die Korridore zu empfehlen sind, liegen im Nord-Westen. Dadurch könnten bedeutende Wege für die Wildkatze weiter in den Süden geschaffen werden, wo die Besiedlungswahrscheinlichkeit in der Simulation niedrig ausfällt und auch nur vereinzelt Wildkatzennachweise vorhanden sind (s. Abbildung 18). Das erste von diesen beiden Gebieten liegt bei Feschnitz. Hier werden zwei mögliche Korridore vorgeschlagen, die das Habitat `5354` und das Habitat `5601` verbinden könnten (s. Abbildung 19). Das Habitat `5354` wird im Süden allerdings von einer Autobahn abgeschnitten, die im Zusammenspiel mit der dahinter liegenden Landwirtschaft offensichtlich eine effektive Barriere für die Wildkatze darstellt. Um so bedeutender wäre es allerdings auch wenn hier eine Brücke zwischen Habitaten geschlagen werden könnte. Das zweite der Gebiete im Nord-Westen ist recht groß und umfasst zwei vorgeschlagene Korridore: zum einen bei Melk (s. Abbildung 20) und zum anderen bei Getzersdorf (s. Abbildung 21). Bei Melk sollen die Habitate `18623` und `3746` verbunden werden. Dieses Teilgebiet ist aber sehr urban geprägt uns mehrere Hauptstraßen und eine Autobahn müssten gekreuzt werden. Möglicherweise lässt sich aber mit lokaler Expertise auch hier eine Verbindung dieser beiden Habitate finden. Bei Getzersdorf wäre das Ziel des Korridors die Habitate `18623` und `2508` zu verbinden. Dies könnte eine bedeutende Verbindung von dem Nord-Westen mit vielen Wildkatzennachweisen und einer hohen Besiedlungswahrscheinlichkeit nach Osten schaffen. Das zu lösende Problem wäre hierbei die Kreuzung des Korridors mit der Autobahn kurz vor dem Habitat `2508` . Das dritte ausgewählte Gebiet in Niederösterreich liegt bei Matzen. Hier könnten die Habitate `3465`, `3461` und `2490` miteinander verbunden werden (s. Abbildung 22). Dabei könnte das Habitat `3461` als Trittstein Habitat fungieren; allerdings würden auch hier die Korridore Hauptstraßen kreuzen.

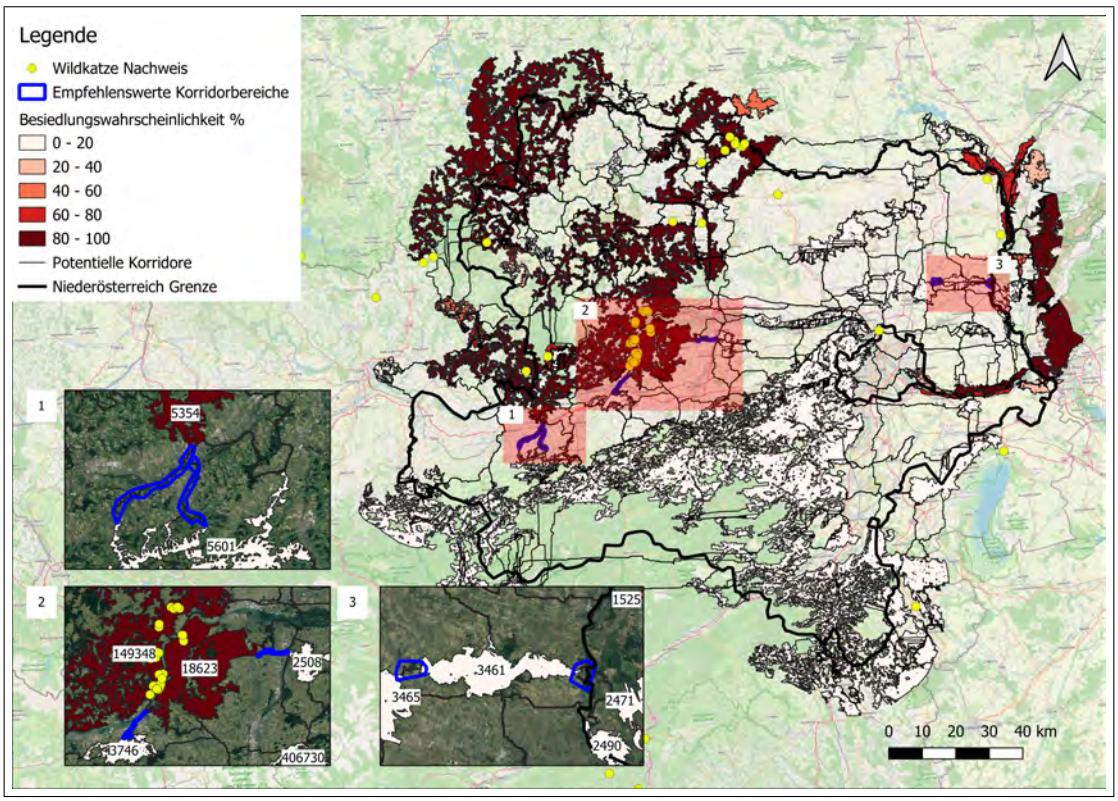


Abbildung 18: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in Niederösterreich.

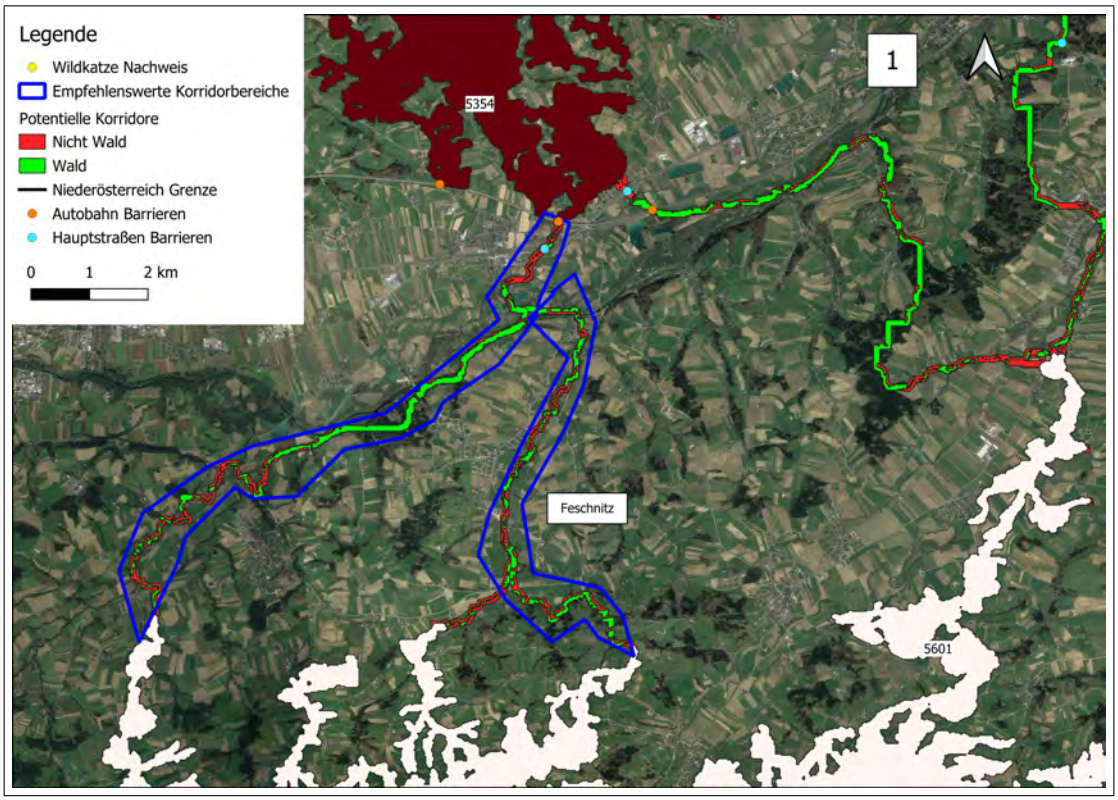


Abbildung 19: Erstes empfohlenes Korridorgebiet in Niederösterreich (s. Abbildung 18).

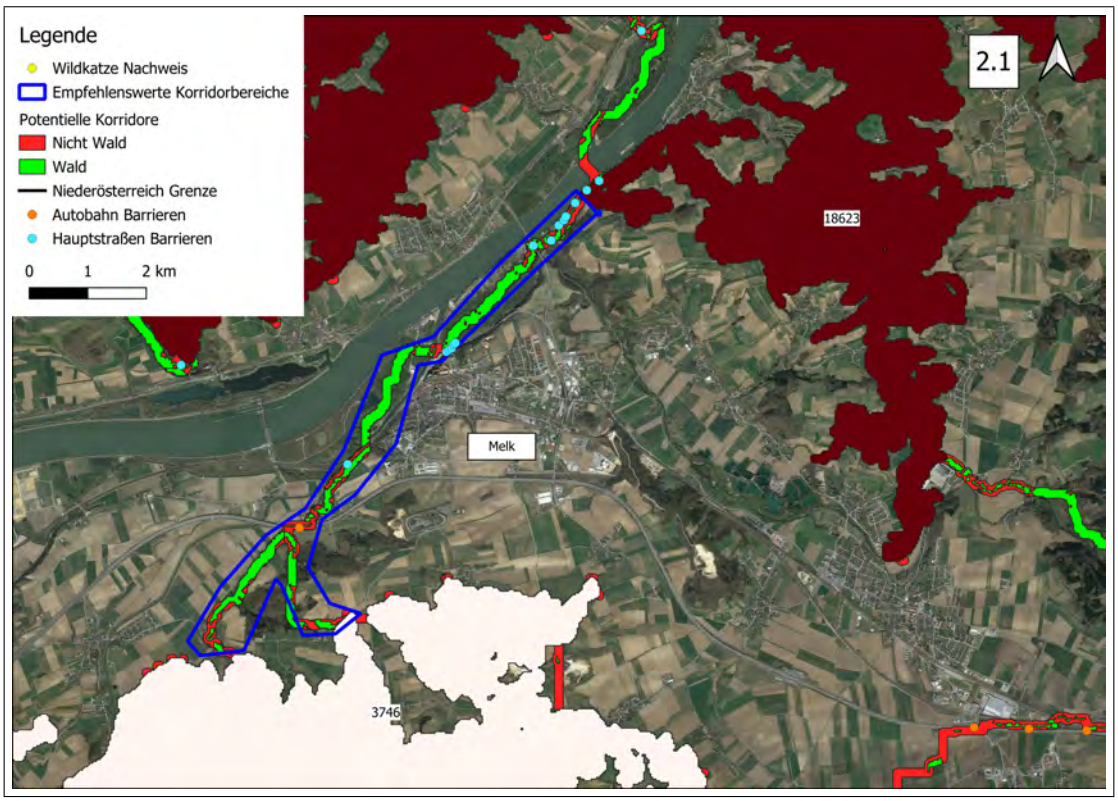


Abbildung 20: Zweites empfohlenes Korridorgebiet in Niederösterreich (s. Abbildung 18).

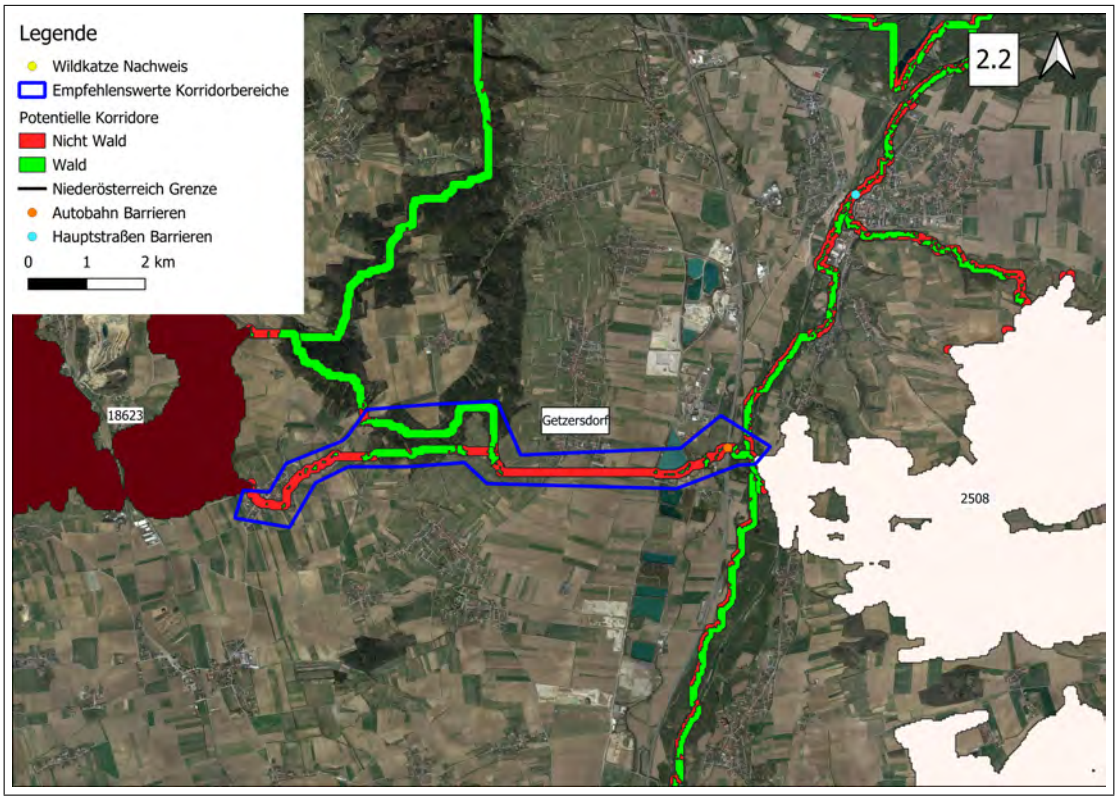


Abbildung 21: Drittes empfohlenes Korridorgebiet in Niederösterreich (s. Abbildung 18).

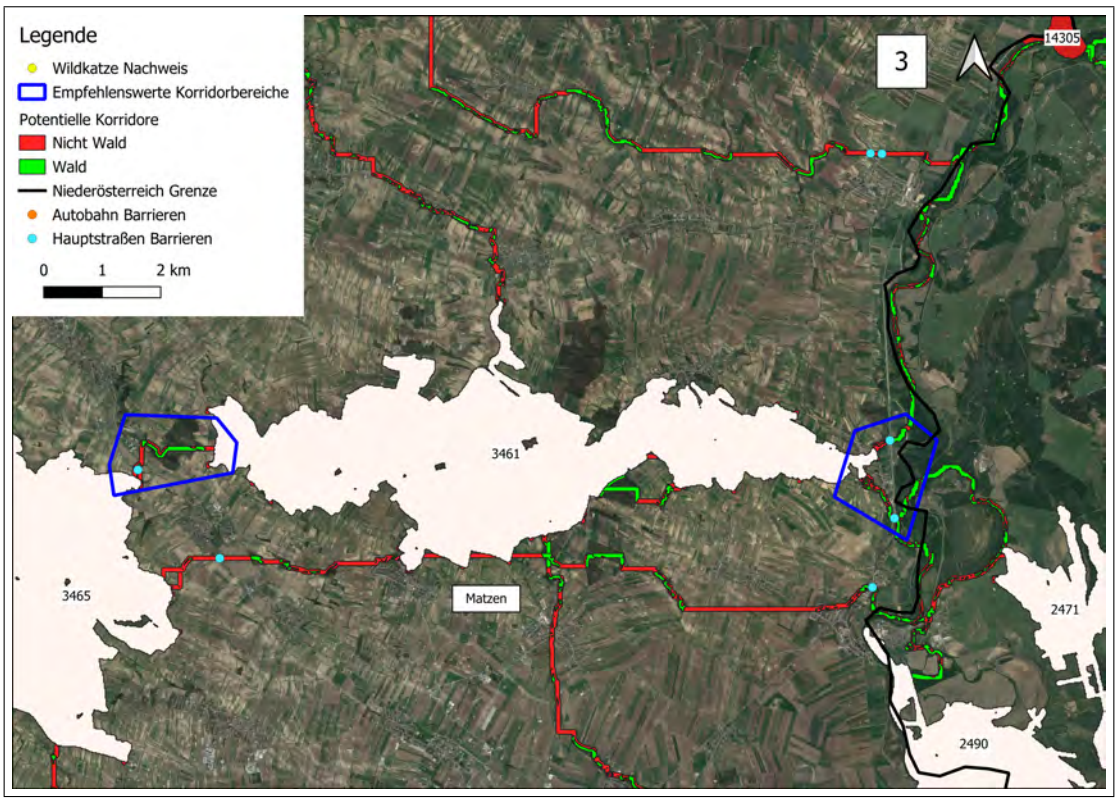


Abbildung 22: Viertes empfohlenes Korridorgebiet in Niederösterreich (s. Abbildung 18).

3.2.5 Oberösterreich

Hier befinden sich die Gebiete mit einer höheren Besiedlungswahrscheinlichkeit in der Simulation an der nord-östlichen Grenze Oberösterreichs. Die durchschnittliche Besiedlungswahrscheinlichkeit für die Habitate in Oberösterreich liegt bei 7 %. Allerdings sind auch einige Nachweise für Wildkatzen an der nördlichen Grenze vorhanden, wo die Simulation eine geringere Besiedlungswahrscheinlichkeit ergeben hat. Die beiden empfohlenen Korridorgebiete liegen trotzdem orientiert an den simulierten Besiedlungswahrscheinlichkeit im Nordwesten Oberösterreichs; das Einrichten von Korridoren dort könnte möglicherweise die Besiedlungswahrscheinlichkeit in den angrenzenden Habitaten im Norden erhöhen, in denen die simulierte Besiedlungswahrscheinlichkeit deutlich geringer ist (s. Abbildung 23). Das erste ausgewählte Gebiet befindet sich bei Wolfgrub. Hier könnten die Habitate `2753` und `9354` miteinander verbunden werden. Für den Korridor könnten bereits bestehende Waldflächen zwischen den Habitaten genutzt werden. Eine Barriere durch eine größere Straße wäre nicht vorhanden. Diese Gegebenheiten scheinen die Planung eines Korridors durchaus zu begünstigen. Allerdings würde sich auch ein Blick auf potentielle Verbindungsmöglichkeiten zwischen den Habitaten `14659` und `2753` lohnen, da diese beiden durch eine größere Straße getrennt voneinander sind. Wenn hier Maßnahmen ergriffen werden können die die Barrieren kleiner lassen werden, würde das folglich auch die Funktion des Korridors zwischen den Habitaten `2753` und `9354` begünstigen (s. Abbildung 24). Das zweite ausgewählte Gebiet befindet sich bei Sankt Georgen an der Guzen. Hier könnten zum einen potentiell die Habitate `14659` und `9354` und zum anderen die Habitate `14659` und `2590` miteinander verbunden werden. Bei dem potentiellen Korridor zwischen den Habitaten `14659` und `9354` scheint vor allem die Barriere durch die Straße kurz vor dem Habitat `9354` die Herausforderung zu sein. Bei der Verbindung zwischen `14659` und `2590` ist eine weitere Strecke zu überwinden, allerdings könnten manche bereits bestehende Waldstücke genutzt werden. Allerdings gilt es auch hier die Barriere durch die Straße bei dem Habitat `14659` zu überwinden (s. Abbildung 25).

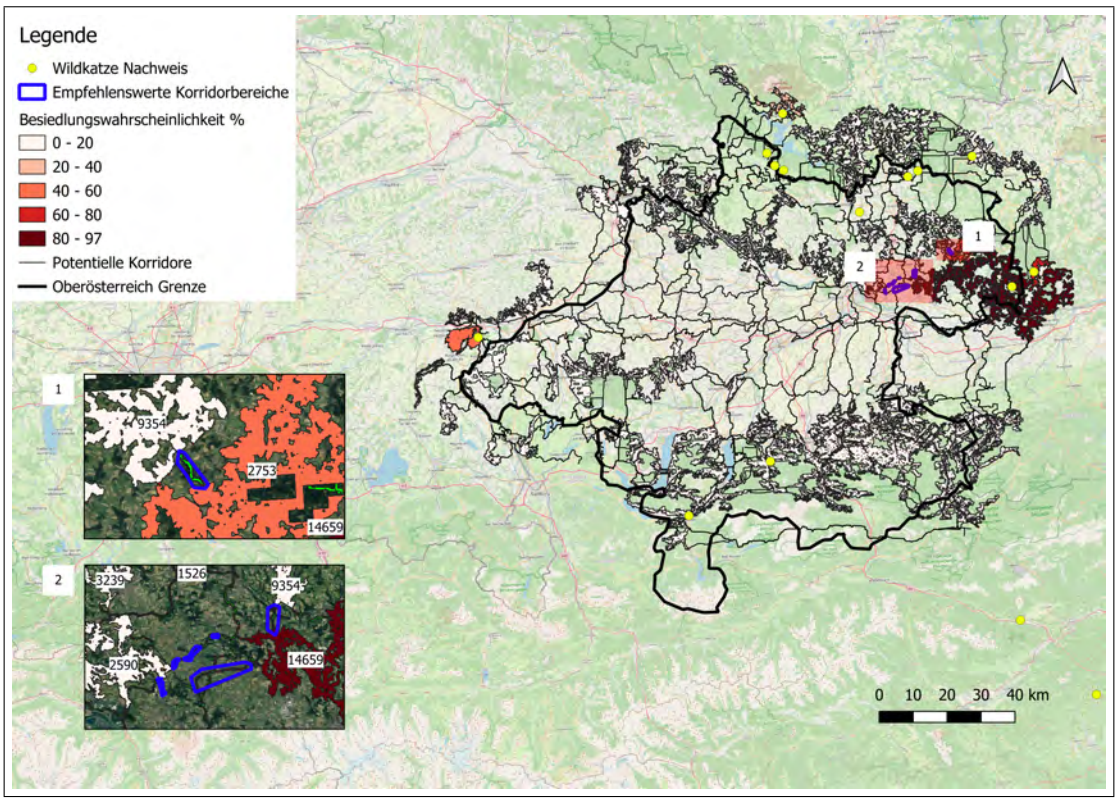


Abbildung 23: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in Oberösterreich.

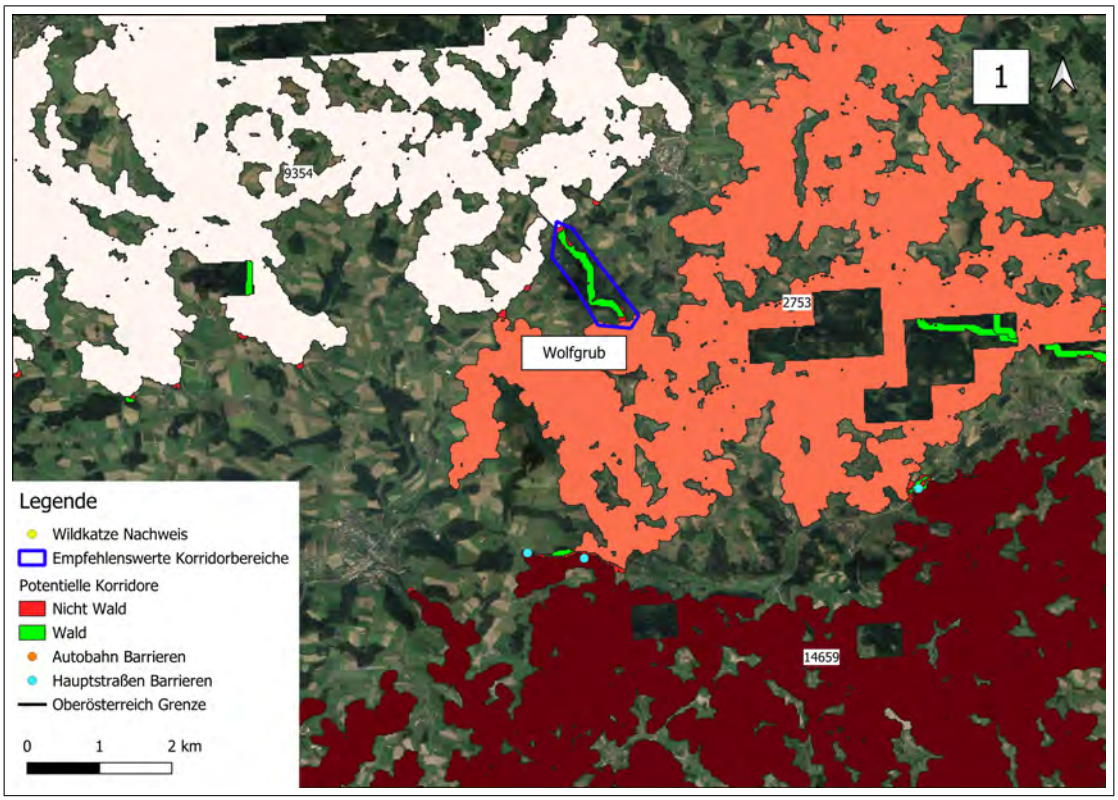


Abbildung 24: Erstes empfohlenes Korridorgebiet in Oberösterreich (s. Abbildung 23).

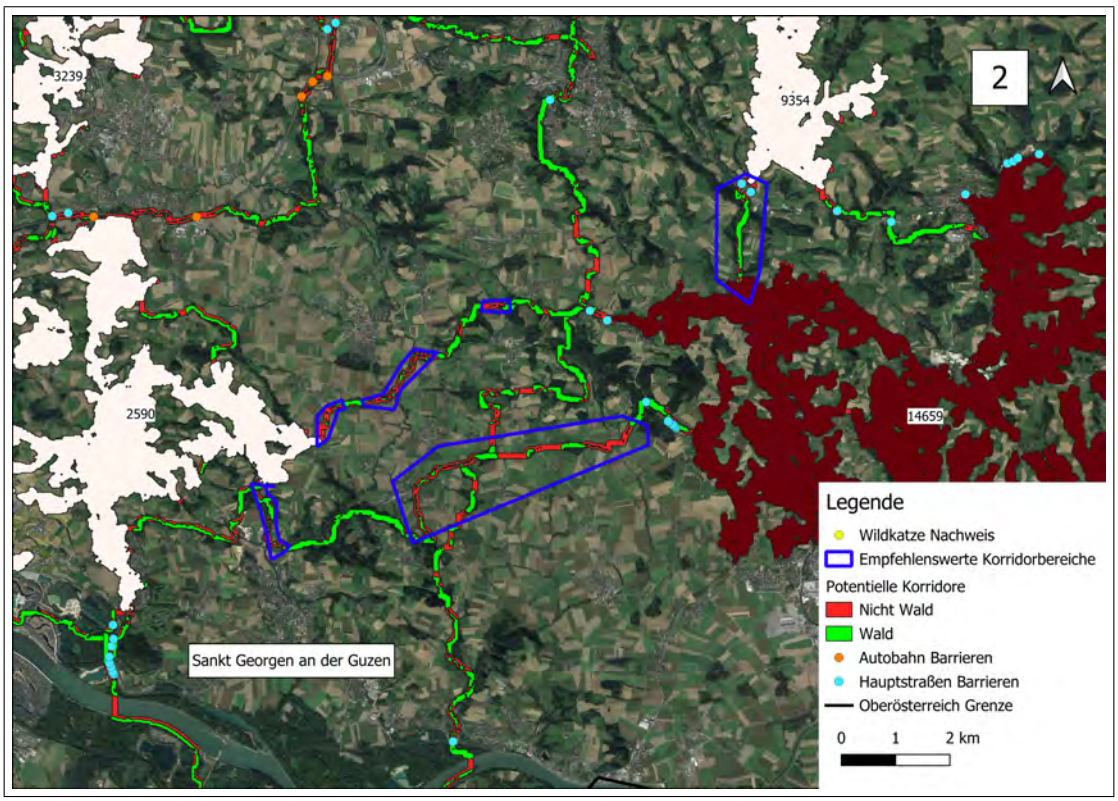


Abbildung 25: Zweites empfohlenes Korridorgebiet in Oberösterreich (s. Abbildung 23).

3.2.6 Salzburg

Für Salzburg sind relativ wenige Habitate verzeichnet und auch ausschließlich in der nördlichen Hälfte des Bundeslandes. Die simulierte mittlere Besiedlungswahrscheinlichkeit für die Gesamtheit der Habitate liegt bei 10 %. Die Habitate mit einer höheren Besiedlungswahrscheinlichkeit befinden sich vor allem im Norden Salzburgs. Zwei der vorgeschlagenen Korridorgebiete liegen dementsprechend im Norden. Ein drittes befindet sich weiter im Süden und zeigt eine besondere Herausforderung auf einem besonders langen Korridor (s. Abbildung 26). Das erste ausgewählte Korridorgebiet im Norden liegt bei Oberndorf bei Salzburg. Hier könnten die automatisch generierten Korridore eine Orientierung bieten um das vorhandene potentielle Habitat `2559` zu erweitern und zukünftig möglicherweise mit anderen zu verbinden; vor allem wenn die Schneebedeckung in dem kommenden Jahren abnehmen sollte (s. Abbildung 29). Das zweite Korridorgebiet in Norden liegt bei Unternach. Hier besteht die spezielle Herausforderung darin die Habitate `6163` und `2183` über das urbane Gebiets Unternachs zu verbinden, wobei die angrenzenden Gewässer einen natürlichen Engpass bilden (s. Abbildung 28). Das dritte Korridorgebiet zeigt die spezielle Herausforderung wenn natürliche und durch Menschen geschaffene Barrieren zusammenkommen. Hier müsste eine Lösung zur Überbrückung von urbanen Gebiet sowie des Flusses gefunden werden um den Weg des Korridors weiter gen Süden zu führen (s. Abbildung 27).

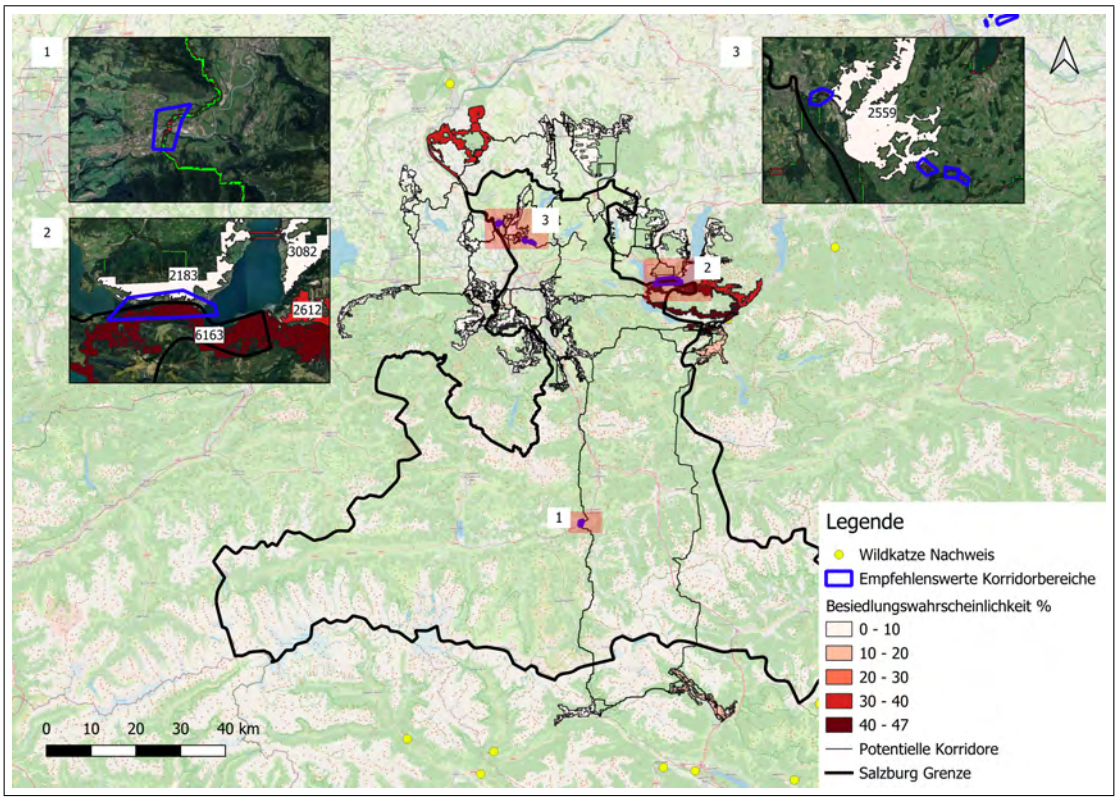


Abbildung 26: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in Salzburg.

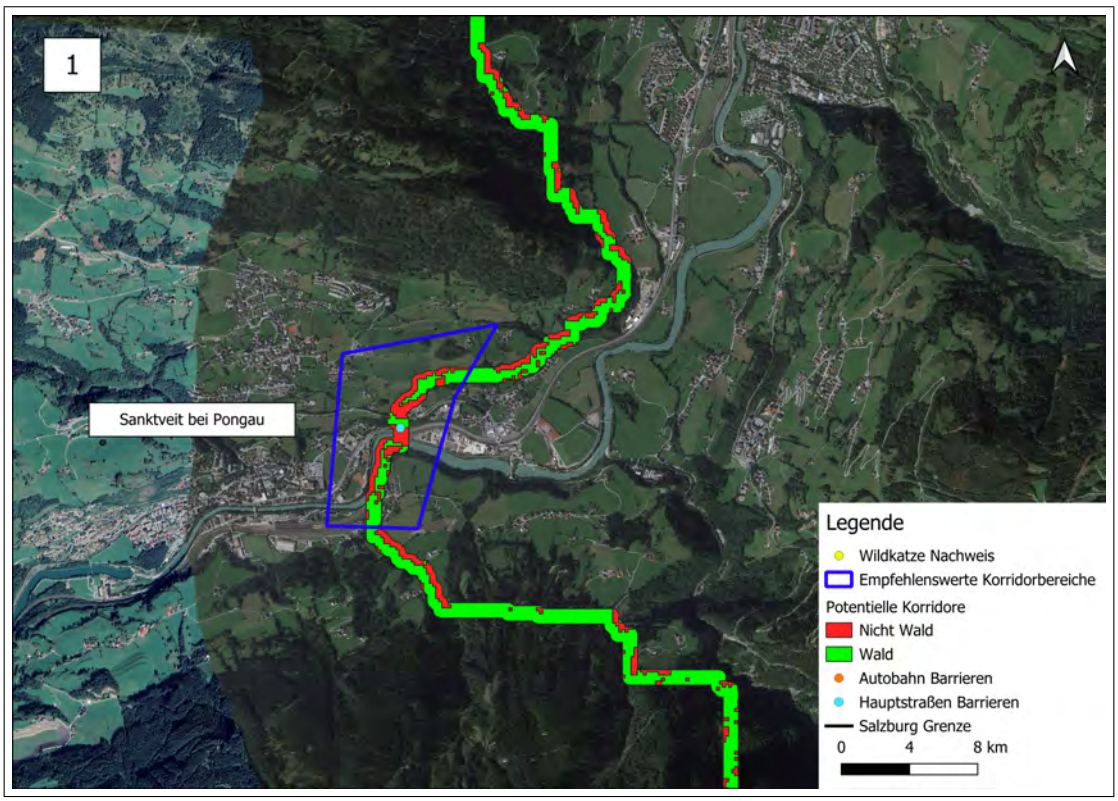


Abbildung 27: Erstes empfohlenes Korridorgebiet in Salzburg (s. Abbildung 26).

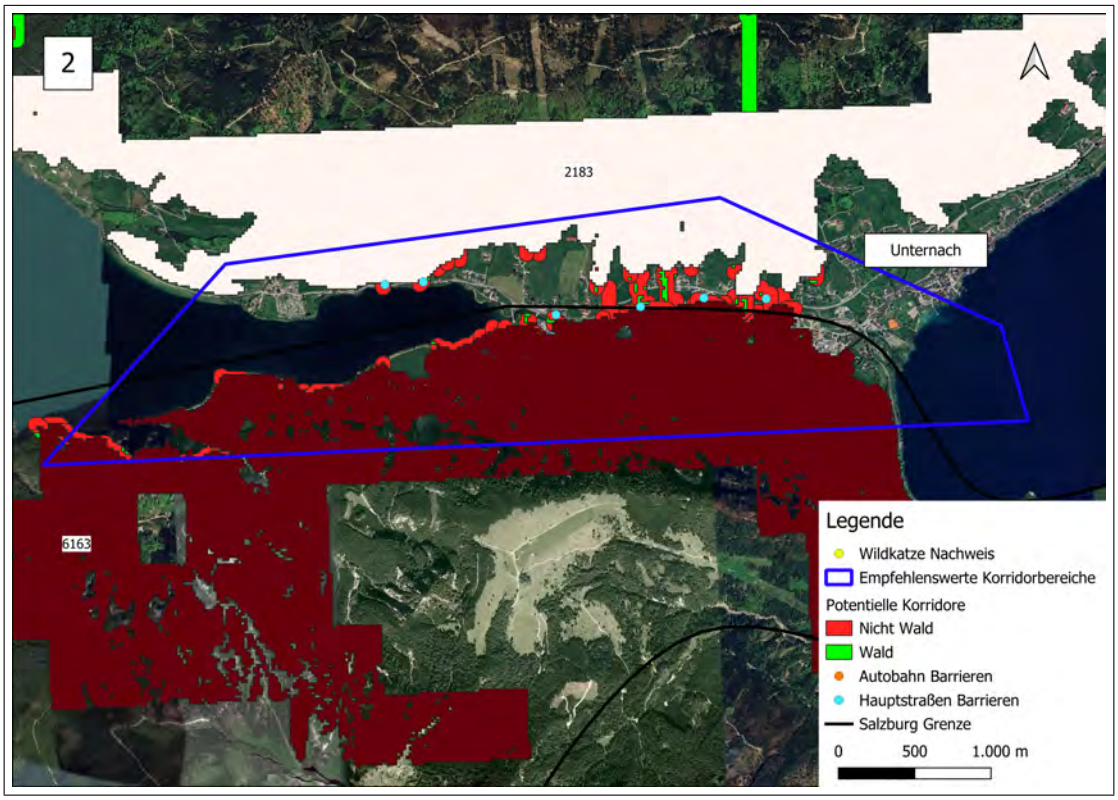


Abbildung 28: Zweites empfohlenes Korridorgebiet in Salzburg (s. Abbildung 26).

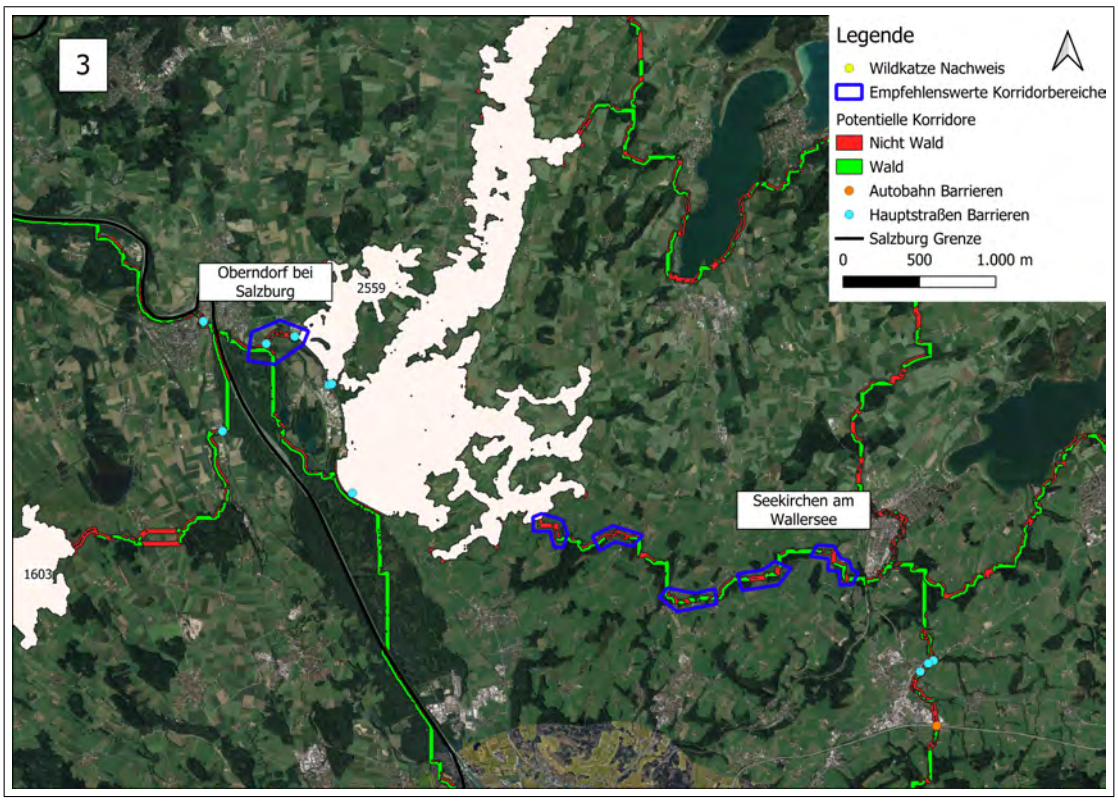


Abbildung 29: Drittes empfohlenes Korridorgebiet in Salzburg (s. Abbildung 26).

3.2.7 Vorarlberg und West Tirol

Auch in Vorarlberg und West Tirol sind relativ wenige Wildkatzen Habitate zu finden; diese befinden sich hauptsächlich in den größeren Tälern und sind dadurch besonders lang gestreckt. Die simulierte durchschnittliche Besiedlungswahrscheinlichkeit beträgt für die Habitate 6 %. Die beiden ausgewählten Gebiete mit potentiellen Korridoren befinden sich im Westen Nahe der Grenze, wo auch höhere Besiedlungswahrscheinlichkeiten zu finden sind (s. Abbildung 30). Das erste Gebiet bei Frastanz enthält einen vorgeschlagenen Korridor, der die Habitate `6601` und `2391` verbinden könnte. Allerdings stellt hier die Autobahn eine bedeutende Barriere dar, wobei der Korridor scheinbar auch ohne Kreuzen entlang der Autobahn geführt werden könnte. Trotzdem könnte das Kreuzen der Autobahn auch Vorteile haben. Die exakten Möglichkeiten und Einschränkungen müssten hier lokal erörtert werden (s. Abbildung 31). Das zweite Gebiet liegt bei Triesenberg. Hier könnten die Habitate `2075`, `3253` und `1779` miteinander verbunden werden. Auch hier stellen Autobahnen, Hauptstraßen und Fluss Barrieren dar, insbesondere zwischen Habitat `2075` und Habitat `3253`. Der Korridor zwischen `3253` und `1779` könnte den Luftbildern zufolge ohne Barrieren und unter Einbeziehen bestehender Waldstücke etabliert werden (s. Abbildung 32).

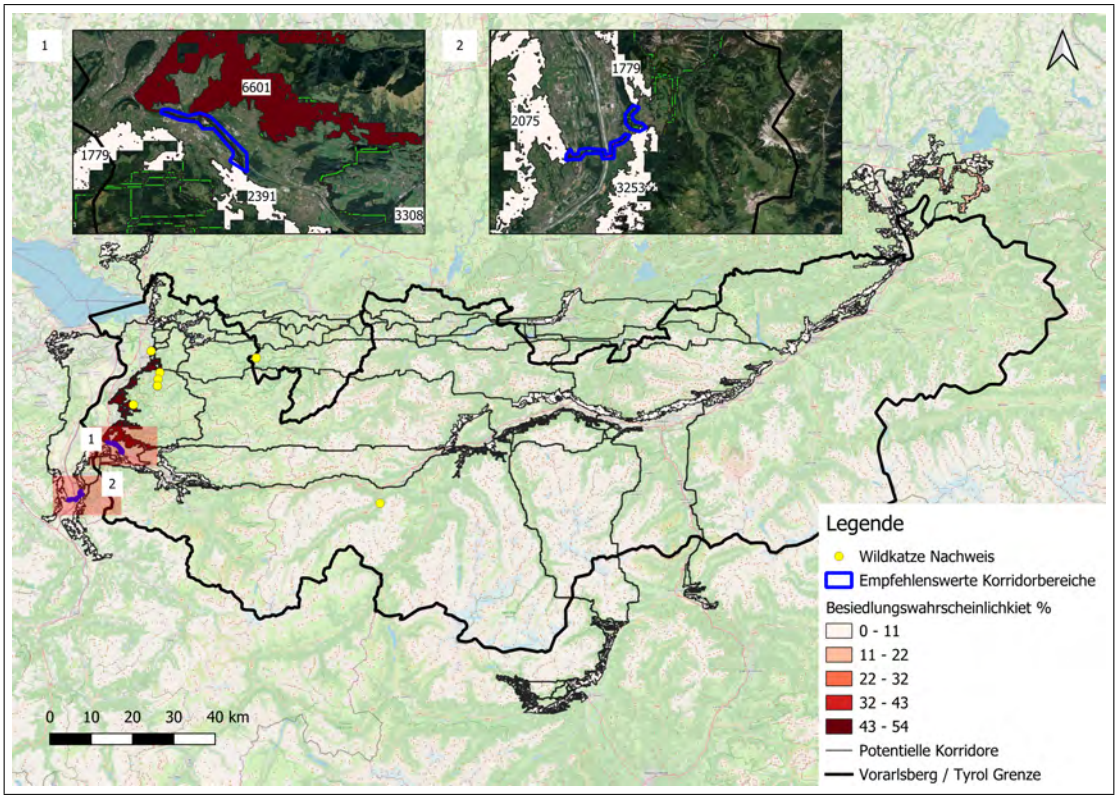


Abbildung 30: Habitat Besiedlungswahrscheinlichkeit und potentielle Korridorflächen in Vorarlberg und West Tirol.

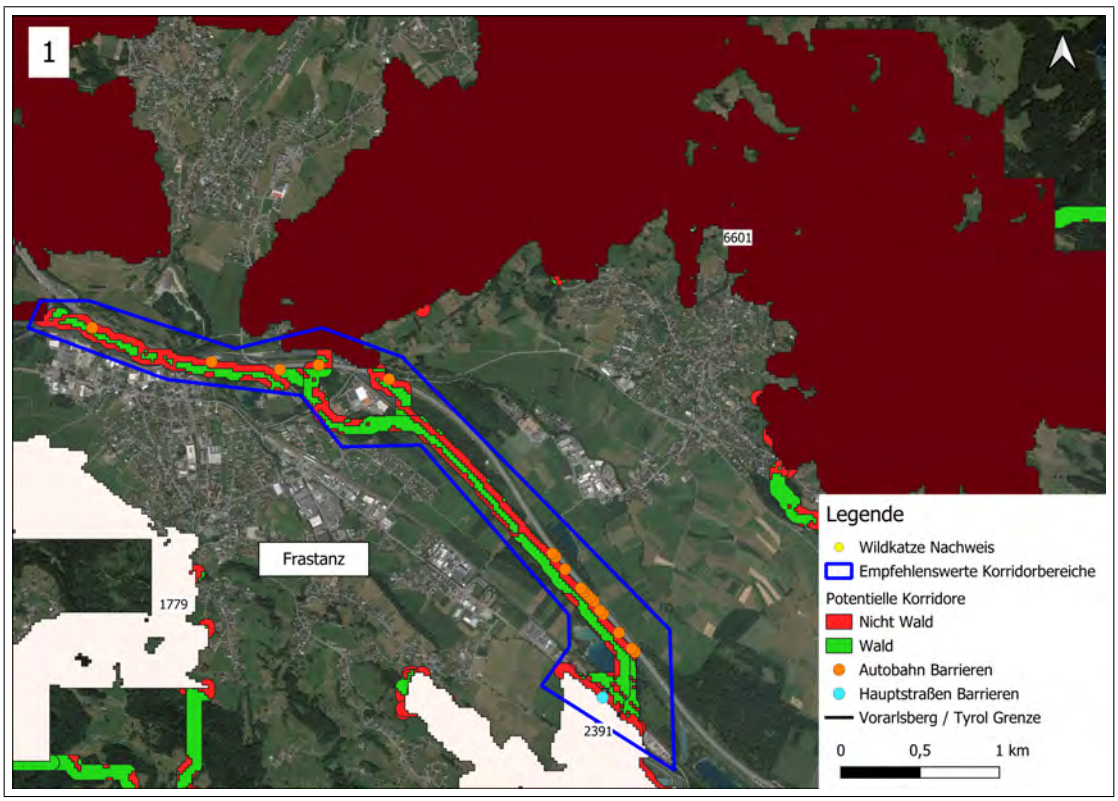


Abbildung 31: Erstes empfohlenes Korridorgebiet in Vorarlberg und West Tirol (s. Abbildung 30).

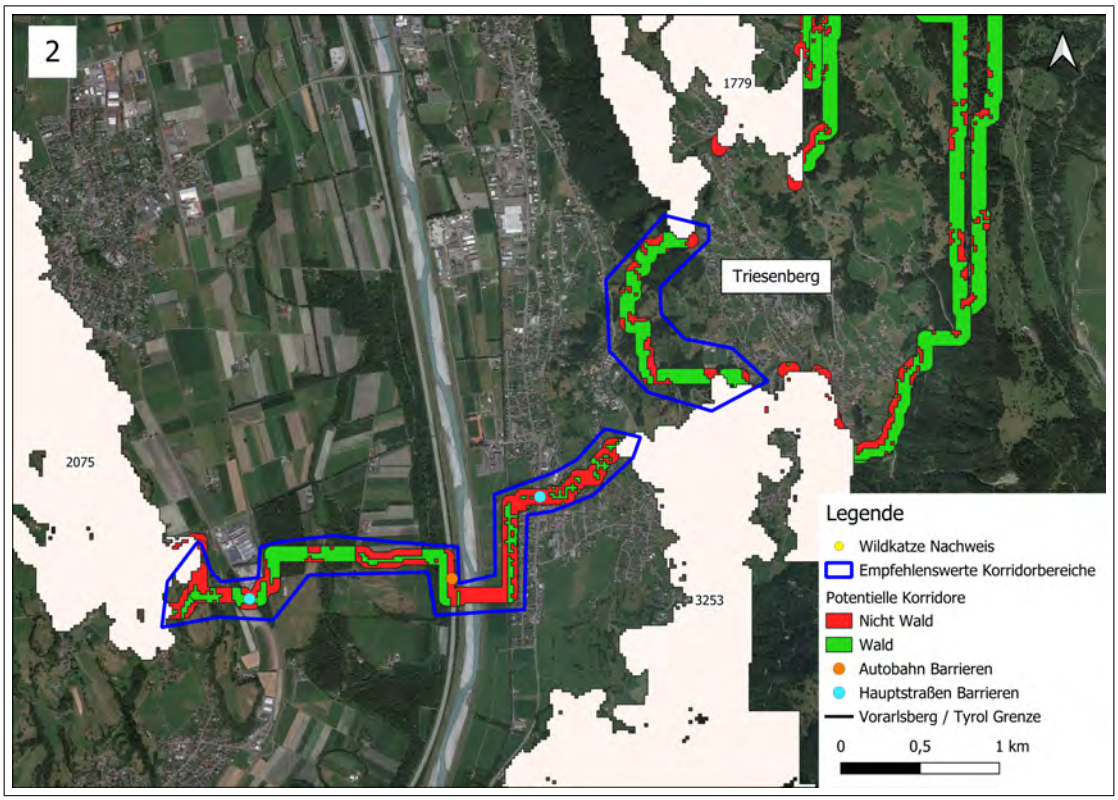


Abbildung 32: Zweites empfohlenes Korridorgebiet in Vorarlberg und West Tirol (s. Abbildung 30).

4 Diskussion

Die Ergebnisse zeigen eine Zunahme der Wildkatzenverbreitung in fast allen Bundesländern mit Ausnahme von Salzburg. Die Ergebnisse sind jedoch wohl konservativ Schaätzungen, da die Auswirkungen des Klimawandels vorraussichtlich mehr Lebensräume schaffen, wie in Tabelle 5 zu sehen ist. Darüber hinaus umfasst die Habitatanalyse nur geeignete landwirtschaftliche Lebensräume, die innerhalb der Pufferzone von Waldlebensräumen liegen. Studien haben aber gezeigt, dass Wildkatzen in der Lage sind, ihr Revier fast ausschließlich in landwirtschaftlich genutzten Lebensräumen zu errichten (Jerosch et al., 2017) auch diese Möglichkeit ist in unserer Habitatanalyse nicht berücksichtigt, was sich auf die Dichte und Geschwindigkeit der Besiedlung neuer Lebensräume auswirken könnte. Die in Bouerdoux et al. (2019) verwendeten Kostenwerte für die Landwirtschaft waren die höheren der beiden in dieser Arbeit verwendeten Werte, was ebenfalls als konservativ angesehen werden kann. Wie bei jedem Modell handelt es sich auch hier nur um eine Vorhersage auf der Grundlage von Daten und nicht um eine zwangsläufige Vorhersage für die Zukunft. Es sind weitere Forschungen zur Ausbreitung der Wildkatze erforderlich, um die Modelle genauer zu verbessern, aber auch dann wird ein Modell immer ein Modell bleiben und mit Unsicherheiten arbeiten. Die Einbeziehung potenzieller Gebiete, in denen die Wildkatze bisher nicht gefunden wurde, ist wichtig, damit in diesen Gebieten Entscheidungen getroffen werden können, falls die Vorhersagen falsch sind. Das Vorkommen der Wildkatze in der Wachau weit entfernt von der nächstgelegenen Population in Bayern ist ein gutes Beispiel für die Unvorhersehbarkeit des Wildkatzenvorkommens in Österreich. Es ist möglich, dass die Population in der Wachau nie erloschen ist, und vielleicht gibt es solche Populationen auch anderswo in Österreich. Es ist wichtig, die Ergebnisse der Habitatbesetzung im Laufe der Zeit eher als Indikator für die Vernetzung zu betrachten, wobei die Empfehlungen für den Korridor nur auf der Grundlage der aktuellen Verbreitung erfolgen. Der zur Erstellung des Wildkatzenkorridors verwendete Algorithmus verwendet ein Modell der geringsten Kosten, das die Subjektivität des menschlichen Benutzers vernachlässigt. Der Maßstab von 20 m^2 ist nicht 100% genau, daher ist auch eine Sichtung in der Praxis erforderlich, um festzustellen, ob ein potenzieller Korridor Unterschiede zu den Karten aufweist. Die Qualität der Lebensräume kann innerhalb derselben Lebensraumklasse sehr unterschiedlich sein. Daher wird eine individuelle Bewertung der Lebensräume empfohlen, um zu verhindern, dass Verallgemeinerungen zu Fehlern führen. Darüber hinaus sollten bei der Entscheidung, ob ein Korridor angelegt oder aufgewertet werden soll, auch Aspekte wie Topographie, Arbeitsaufwand für die Anpflanzung, wirtschaftliche Kosten und soziale Kosten berücksichtigt werden. Die vorgeschlagenen Korridorflächen würden unweigerlich Ausbreitungswege nicht nur für die Wildkatze, sondern auch für viele andere Arten bieten. Lokales Wissen über die Ökologie könnte daher bei der Festlegung von Prioritäten hilfreich sein und die Tür für Kooperationsprojekte öffnen. Es ist zwar zu hoffen, dass zumindest einige der Korridore realisierbar sind, doch können Korridore, die es nicht sind, zumindest nützlich sein, um umfassendere Probleme der Lebensraumvernetzung aufzuzeigen

oder auch Gebiete, die erhalten werden müssen. Wenn die Korridore umgesetzt werden, wäre es auch zusätzlich möglich, RangeshifteR zu verwenden, um die Korridore auf Unterschiede vor und nach der Schaffung des Korridors zu testen. Wenn die Ressourcen begrenzt sind, könnte dies bei der Entscheidung helfen, welche Korridore am effektivsten zu bepflanzen sind, um den größten Effekt zu erzielen.

4.1 Korridor vergleich Niederösterreich

Im Jahr 2020 wurde in Niederösterreich ein Wildkatzenkorridormodell von (Leissing et al.) durchgeführt. In diesem Papier wurden Bereiche vorgeschlagen, in denen an der Schaffung von Korridoren gearbeitet werden sollte. Abbildung 34 vergleicht unsere beiden Studienkorridore. Vier der Korridore überschneiden sich vollständig, während die anderen denselben Lebensraum miteinander verbinden. Das dargestellte Habitatgebiet ist 149348 Hektar groß, seine Konnektivität variiert jedoch. Das Gebiet zwischen Eggenberg und Horn stellt die einzige Verbindung zwischen den Wildkatzen im Thayatal und jenen in der Wachau dar.

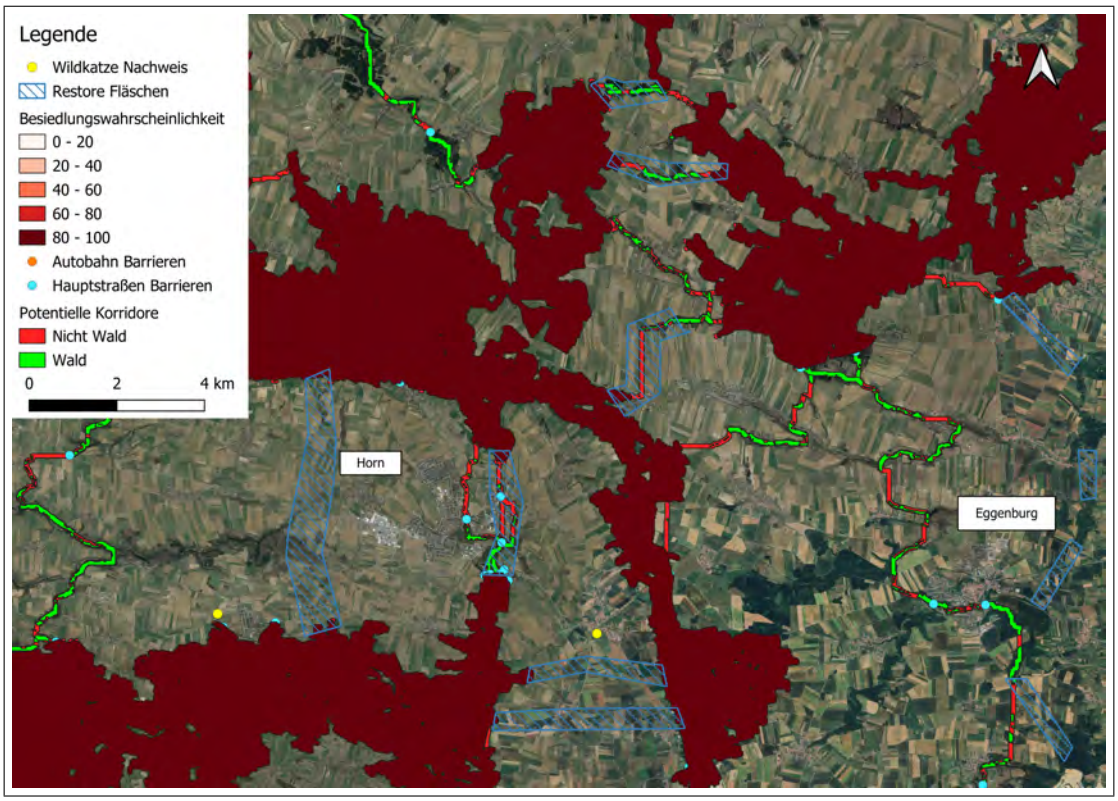


Abbildung 33: Vorgeschlagene Restore Fläschen aus Leissing et al. im Vergleich zu unseren Studienkorridoren.

Südlich der Wachau gibt es große Lebensraumflächen, wobei das große zusammenhängende Gebiet 406730 Hektar groß ist. In unserer Studie wurden Korridore ausgewählt, die die Wachau mit Lebensräumen im Süden verbinden, mit der Begründung, dass es hier überhaupt keine Korridore gibt, obwohl das Potenzial ebenfalls groß ist. Ideal wäre es natürlich, wenn in beiden

Gebieten Korridore geschaffen würden, wobei es vielleicht als wichtiger angesehen werden könnte, das Gebiet zwischen Eggenberg und Horn zu stärken, weil hier Wildkatzen nachgewiesen worden sind.

Literatur

Bastianelli, M. L., Premier, J., Herrmann, M., Anile, S., Monterroso, P., Kuemmerle, T., Dormann, C. F., Streif, S., Jerosch, S., Götz, M., et al. (2021). Survival and cause-specific mortality of European wildcat (*Felis silvestris*) across Europe. *Biological Conservation*, 261:109239.

Bocedi, G., Palmer, S. C., Malchow, A.-K., Zurell, D., Watts, K., and Travis, J. M. (2021). Rangeshifter 2.0: an extended and enhanced platform for modelling spatial eco-evolutionary dynamics and species' responses to environmental changes. *Ecography*, 44(10):1453–1462.

Bocedi, G., Palmer, S. C., Pe'er, G., Heikkinen, R. K., Matsinos, Y. G., Watts, K., and Travis, J. M. (2014). Range s hifter: a platform for modelling spatial eco-evolutionary dynamics and species' responses to environmental changes. *Methods in Ecology and Evolution*, 5(4):388–396.

Bourdouxhe, A., Duflot, R., Radoux, J., and Dufrêne, M. (2020). Comparison of methods to model species habitat networks for decision-making in nature conservation: The case of the wildcat in southern belgium. *Journal for Nature Conservation*, 58:125901.

Dominguez Almela, V., Palmer, S. C., Gillingham, P. K., Travis, J. M., and Britton, J. R. (2020). Integrating an individual-based model with approximate bayesian computation to predict the invasion of a freshwater fish provides insights into dispersal and range expansion dynamics. *Biological Invasions*, 22:1461–1480.

Friembichler, S. and Slotta-Bachmayr, L. (2013). Potential habitats for the European Wildcat (*Felis silvestris silvestris*, SCHREBER 1777) in Austria–a basis for further steps in conservation. *Wetlands*, 1:0.

Geofabrik GmbH and OpenStreetMap Contributors (2023). OpenStreetMap for Austria. `https://download.geofabrik.de/europe/austria.html` Accessed on September 13, 2023.

Gerngross, P., Ambarli, H., Angelici, F., Anile, S., Campbell, R., Ferreras, P., Jose María, G., Götz, M., Jerosch, S., Mengüllüoğlu, D., Monterroso, P., and Zlatanova, D. (2021a). *Felis silvestris. The IUCN Red List of Threatened Species 2022: e.T18104985 9A181050999.*

Gerngross, P., Slotta-Bachmayr, L., and Hagenstein, I. (2021b). Ist die Europäische Wildkatze (*Felis silvestris*) zurück in Österreich? *Säugetierkundliche Informationen*, pages 51–62.

Gil-Sánchez, J. M., Barea-Azcón, J. M., Jaramillo, J., Herrera-Sánchez, F. J., Jiménez, J., and Virgós, E. (2020). Fragmentation and low density as major conservation challenges for the southernmost populations of the European wildcat. *PLoS One*, 15(1):e0227708.

Hansson, L., Söderström, L., and Solbreck, C. (1992). The ecology of dispersal in relation to conservation. In Hansson, L., editor, *Ecological Principles of Nature Conservation: Application in Temperate and Boreal Environments*, pages 162–200. Springer US, Boston, MA.

Hartmann, S. A., Steyer, K., Kraus, R. H., Segelbacher, G., and Nowak, C. (2013). Potential barriers to gene flow in the endangered European wildcat (*Felis silvestris*). *Conservation Genetics*, 14(2):413–426.

Howard-McCombe, J., Jamieson, A., Carmagnini, A., Russo, I.-R. M., Ghazali, M., Campbell, R., Driscoll, C., Murphy, W. J., Nowak, C., O'Connor, T., et al. (2023). Genetic swamping of the critically endangered scottish wildcat was recent and accelerated by disease. *Current Biology*, 33(21):4761–4769.

Jerosch, S., Götz, M., and Roth, M. (2017). Spatial organisation of European wildcats (*Felis silvestris silvestris*) in an agriculturally dominated landscape in Central Europe. *Mammalian Biology*, 82:8–16.

Kikstra, J. S., Nicholls, Z. R., Smith, C. J., Lewis, J., Lamboll, R. D., Byers, E., Sandstad, M., Meinshausen, M., Gidden, M. J., Rogelj, J., et al. (2022). The ipcc sixth assessment report wgiii climate assessment of mitigation pathways: from emissions to global temperatures. *Geoscientific Model Development*, 15(24):9075–9109.

Klar, N., Herrmann, M., Henning-Hahn, M., Pott-Dörfer, B., Hofer, H., and Kramer-Schadt, S. (2012). Between ecological theory and planning practice:(re-) connecting forest patches for the wildcat in lower saxony, germany. *Landscape and Urban Planning*, 105(4):376–384.

Klar, N., Herrmann, M., and KRAMER-SCHADT, S. (2009). Effects and mitigation of road impacts on individual movement behavior of wildcats. *The Journal of Wildlife Management*, 73(5):631–638.

Kramer-Schadt, S., Revilla, E., Wiegand, T., and Breitenmoser, U. (2004). Fragmented landscapes, road mortality and patch connectivity: modelling influences on the dispersal of eurasian lynx. *Journal of Applied Ecology*, 41(4):711–723.

Krofel, M., Južnič, D., and Allen, M. L. (2021). Scavenging and carcass caching behavior by European wildcat (*Felis silvestris*). *Ecological research*, 36(3):556–561.

Lamprecht, A., Semenchuk, P. R., Steinbauer, K., Winkler, M., and Pauli, H. (2018). Climate change leads to accelerated transformation of high-elevation vegetation in the central alps. *New Phytologist*, 220(2):447–459.

Lange, U. and Götz, M. (2014). Die wildkatze in sachsen-anhalt. *Landesamt für Umweltschutz Sachsen-Anhalt, Fachbereich Naturschutz*.

Leissing, D., Leitner, H., and Grillmayer, R. Wildkatzenkorridorplan für das wald-und weinviertel in österreich und die kreise südböhmen und südmähren in tschechien.

Malchow, A.-K., Bocedi, G., Palmer, S. C., Travis, J. M., and Zurell, D. (2021). Rangeshiftr: an r package for individual-based simulation of spatial eco-evolutionary dynamics and species' responses to environmental changes. *Ecography*, 44(10):1443–1452.

Maronde, L., McClintock, B. T., Breitenmoser, U., and Zimmermann, F. (2020). Spatial capture–recapture with multiple noninvasive marks: An application to camera-trapping data of the european wildcat (felis silvestris) using r package multimark. *Ecology and Evolution*, 10(24):13968–13979.

Nussberger, B., Currat, M., Quilodran, C. S., Ponta, N., and Keller, L. F. (2018). Range expansion as an explanation for introgression in European wildcats. *Biological Conservation*, 218:49–56.

Ovenden, T. S., Palmer, S. C., Travis, J. M., and Healey, J. R. (2019). Improving reintroduction success in large carnivores through individual-based modelling: How to reintroduce eurasian lynx (lynx lynx) to scotland. *Biological Conservation*, 234:140–153.

Piechocki, R. (1990). *Die Wildkatze: Felis silvestris*. Ziemsen.

Pinto, F. A., Bager, A., Clevenger, A. P., and Grilo, C. (2018). Giant anteater (myrmecophaga tridactyla) conservation in brazil: Analysing the relative effects of fragmentation and mortality due to roads. *Biological Conservation*, 228:148–157.

Quilodran, C. S., Nussberger, B., Montoya-Burgos, J. I., and Currat, M. (2019). Hybridization and introgression during density-dependent range expansion: European wildcats as a case study. *Evolution*, 73(4):750–761.

Slotta-Bachmayr, L., Friembichler, S., and Hagenstein, I. (2012). Austria-I. Action plan for the conservation of the European wildcat in Austria (Die Wildkatze (*Felis silvestris*) in Oesterreich-I. Aktionsplan zum Schutz der Europäischen Wildkatze in Oesterreich). *Mitteilungen aus dem Haus der Natur*, 20:57–68.

Spitzenberger, F. (2005). Rote Liste der Säugetiere österreichs (Mammalia). In Zulka, K. P., editor, *Rote Listen gefährdeter Tiere österreichs. Checklisten, Gefährdungsanalysen, Handlungsbedarf. Teil 1: Säugetiere, Vögel, Heuschrecken, Wasserkäfer, Netzflügler, Schnabelfliegen, Tagfalter.*, pages 45–62. Grüne Reihe des Bundesministeriums für Land- und Forstwirtschaft, Umwelt und Wasserwirtschaft Band 14/1 (Gesamtherausgeberin Ruth Wallner), Böhlau, Wien.

Streif, S., Kohnen, A., and Wilhelm, C. (2017). Die Wildkatze in den Rheinauen und am Kaiserstuhl. `https://www.waldwissen.net/de/lebensraum-wald/naturschutz/monitoring/wildkatzen-in-baden-wuerttemberg-1` Accessed on Oktober 10, 2023.

Venter, Z. S. and Sydenham, M. A. (2020). ELC10: European 10 m resolution land cover map 2018 (Version 01) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4407051 Accessed on November 19, 2023.

Westekemper, K., Tiesmeyer, A., Steyer, K., Nowak, C., Signer, J., and Balkenhol, N. (2021). Do all roads lead to resistance? state road density is the main impediment to gene flow in a flagship species inhabiting a severely fragmented anthropogenic landscape. *Ecology and Evolution*, 11(13):8528–8541.

Würstlin, S., Segelbacher, G., Streif, S., and Kohnen, A. (2016). Crossing the Rhine: a potential barrier to wildcat (*Felis silvestris silvestris*) movement? *Conservation genetics*, 17:1435–1444.

5 Appendix

5.1 Parametersatz RangeShifter

Tabelle 6: Benutzte Reproduktionsparameter.

Parameter	Wert
Anteil der männlicher Individuen	0.5
Maximale Haremsgröße	2
Wahrscheinlichkeit der Fortpflanzung in der folgenden Saison	0.9
Anzahl der folgenden Saisons vor folgender Fortpflanzung	0
Anzahl der Lebensabschnitte	3
Maximales Alter	11
Habitatspezifische Stärke der Dichteabhängigkeit der Fruchtbarkeit $(1/b)$ – (Individuen/ha) für Wald, Feld und Landwirtschaft innerhalb einer Pufferzone von 50 m	0.0024
Alle anderen Lebensräume	0

Tabelle 7: Benutzte Population-Entwicklungs Parameter.

Lebensabschnitte	Fruchtbarkeit	Entwicklungswahrscheinlichkeit	Überlebenswahrscheinlichkeit
Männlich 0	0	1	0.63
Weiblich 0	0	1	0.63
Männlich 1	0	1	0.65
Weiblich 1	2.9	1	0.84
Männlich 2	3.7	0	0.85
Weiblich 2	3.7	0	0.88

Tabelle 8: Benutzte Abwanderungs Parameter.

Parameter	Wert
Max. Abwanderungswahrscheinlichkeit von weiblichen Jungtieren	0.4
Max. Auswanderungswahrscheinlichkeit von männlichen Jungtieren	0.8
Max. Abwanderungswahrscheinlichkeit von subadulten Tieren	0
Max. Auswanderungswahrscheinlichkeit von adulten Tieren	0
Alpha (Steigung der dichteabhängigen Funktion)	15
Beta (relativer Wendepunkt der dichteabhängigen Funktion)	0.1

Tabelle 9: Benutzte Parameter für den Stochastic Movement Simulator.

Parameter	Wert
Wahrnehmungsreichweite	900 m
Wahrnehmungsreichweite Methode	2
Richtungsabhängige Beständigkeit	3.35
Speichergröße	1
Ziel-Typ	0

Tabelle 10: Benutzte Besiedlungs Parameter.

Parameter	Wert
Weiblich	Weibchen auf Suche nach geeignetem Habitat + Abhängigkeit von der Dichte
Männlich	Männchen auf Suche nach geeignetem Habitat + Dichteabhängigkeit + Paarungsanforderungen
Maximale Ansiedlungswahrscheinlichkeit (Männchen und Weibchen)	1
AlphaS (Steigung der dichteabhängigen Funktion)	-7
BetaS (relativer Wendepunkt der dichteabhängigen Funktion)	0.5
Maximale Anzahl von Schritten	NA

5.2 Karten der Bundesländer

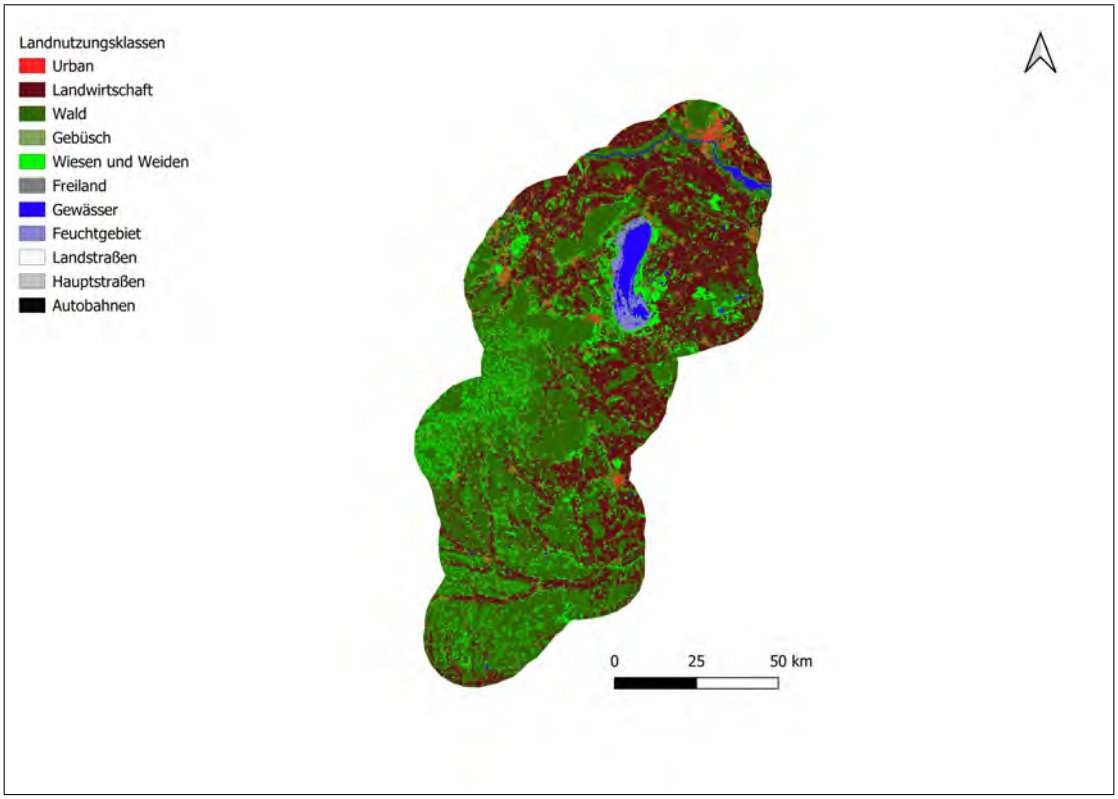


Abbildung 34: Burgenland.

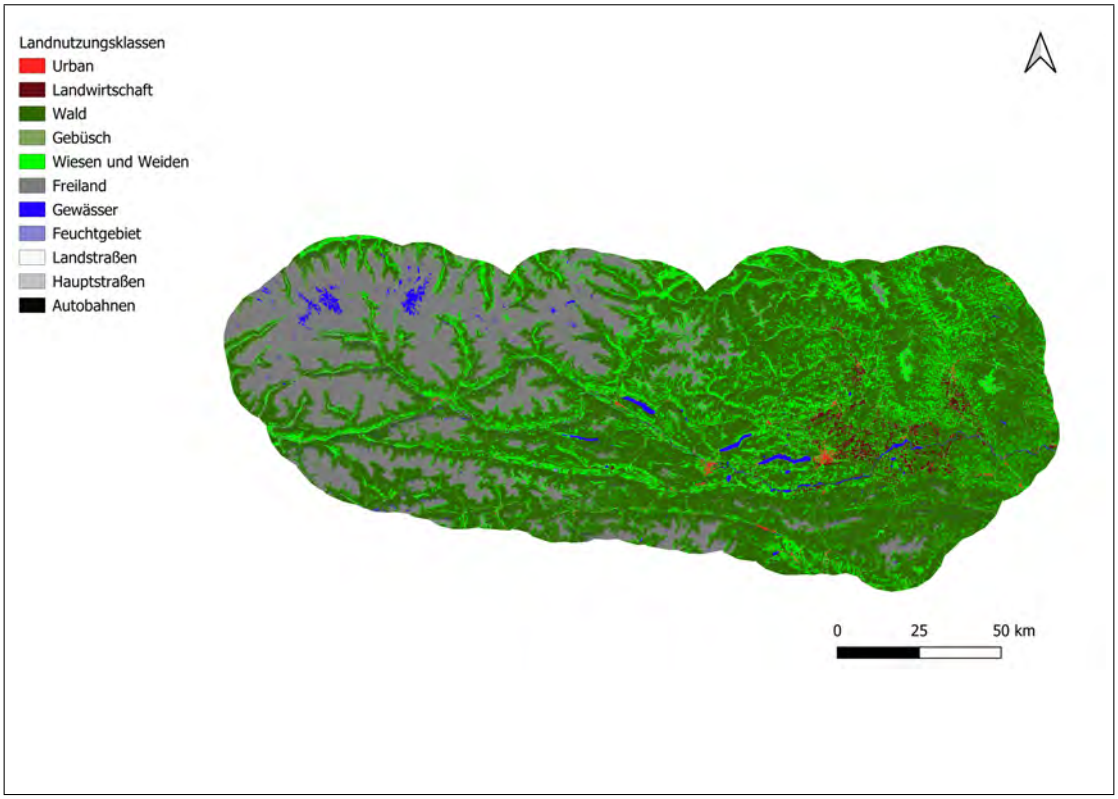


Abbildung 35: Kärnten.

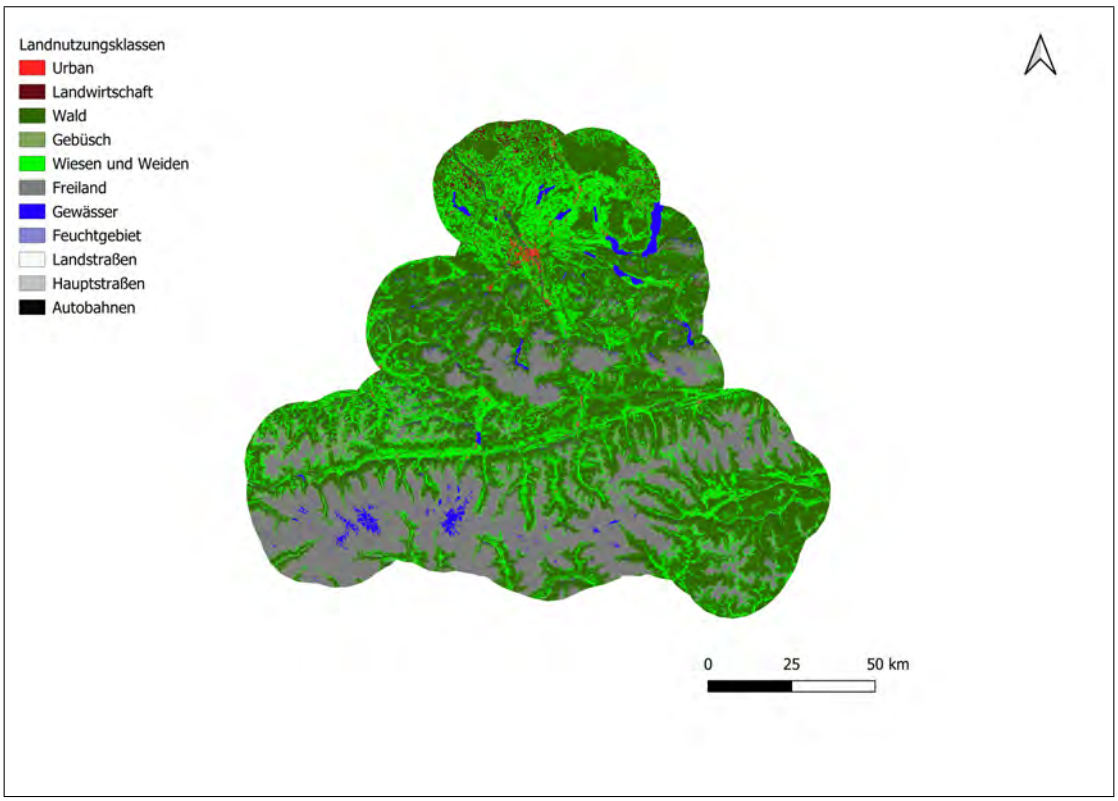


Abbildung 36: Salzburg.

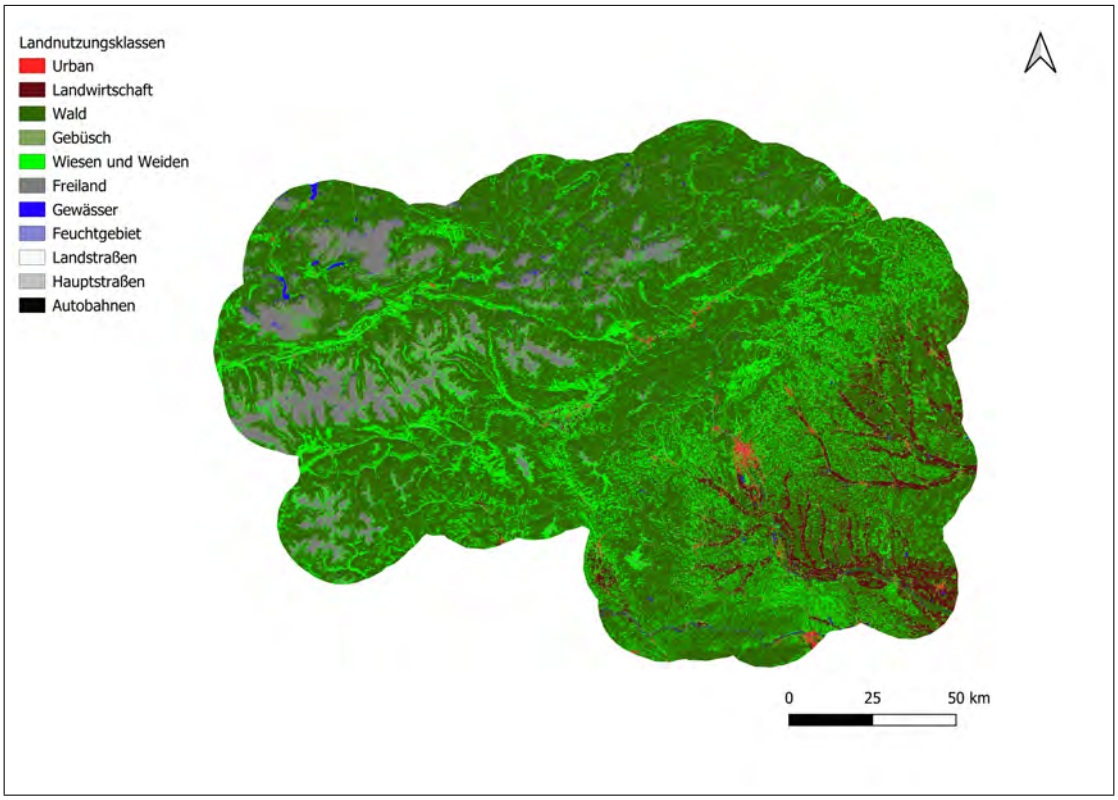


Abbildung 37: Steiermark.

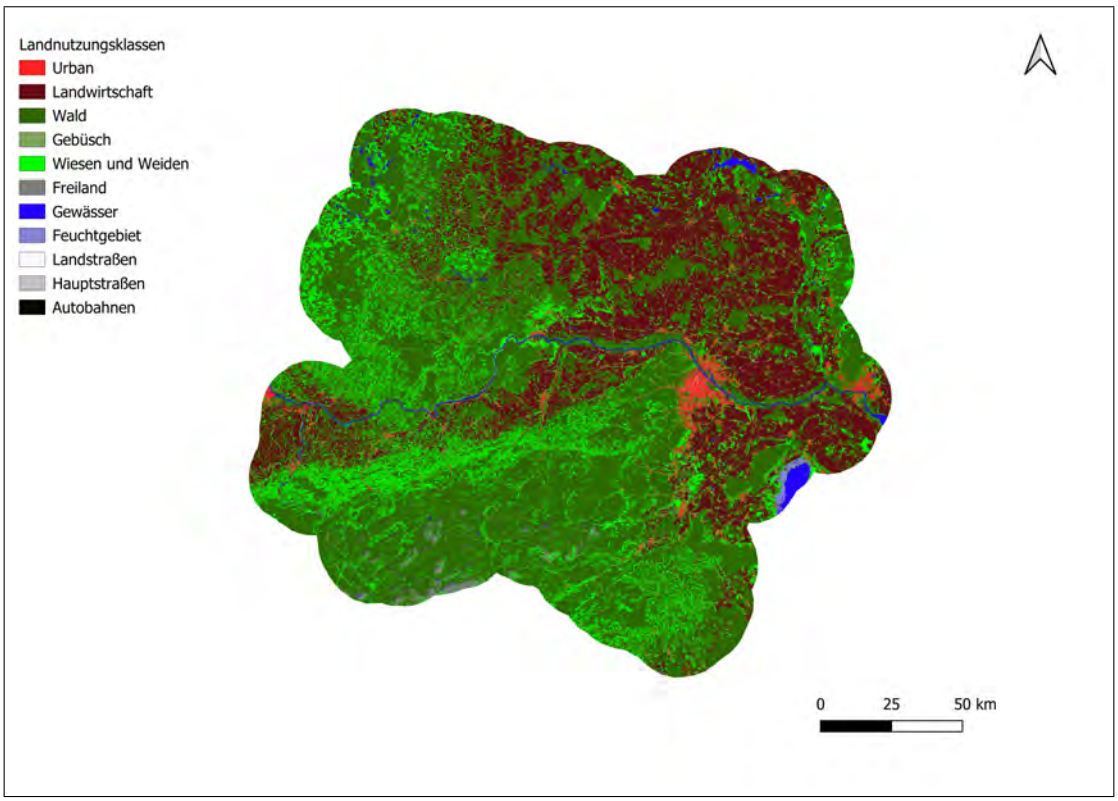


Abbildung 38: Niederösterreich.

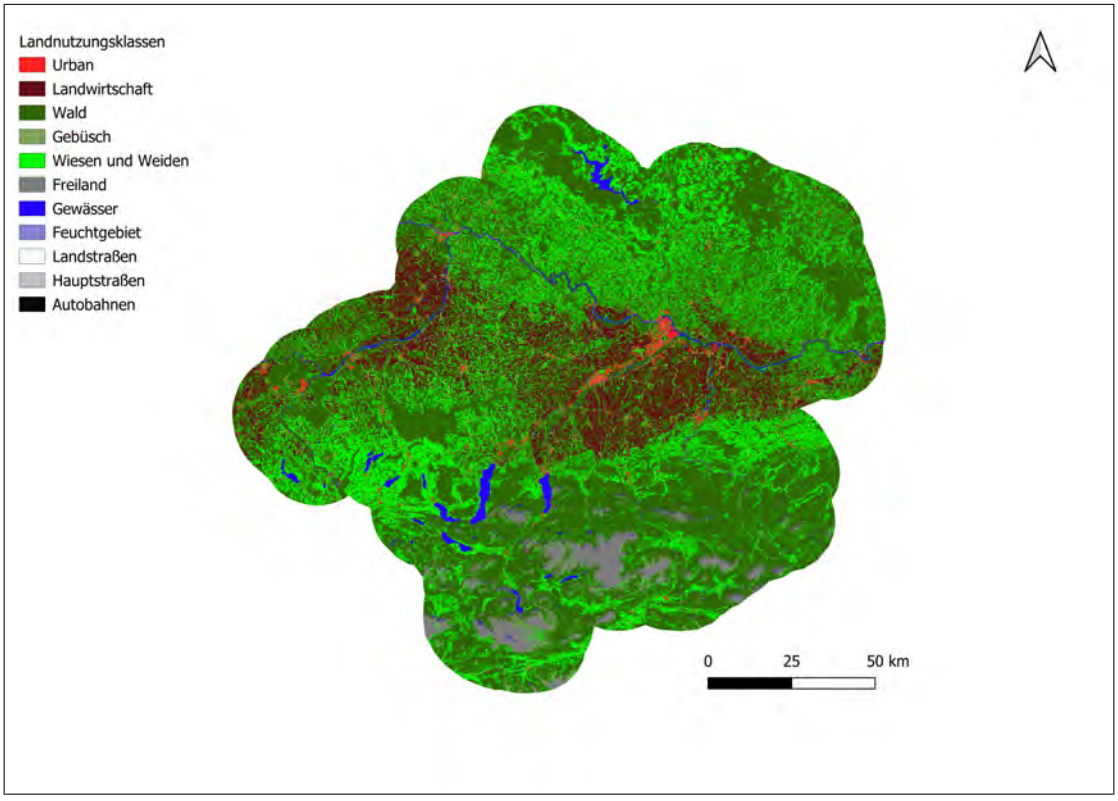


Abbildung 39: Oberösterreich.

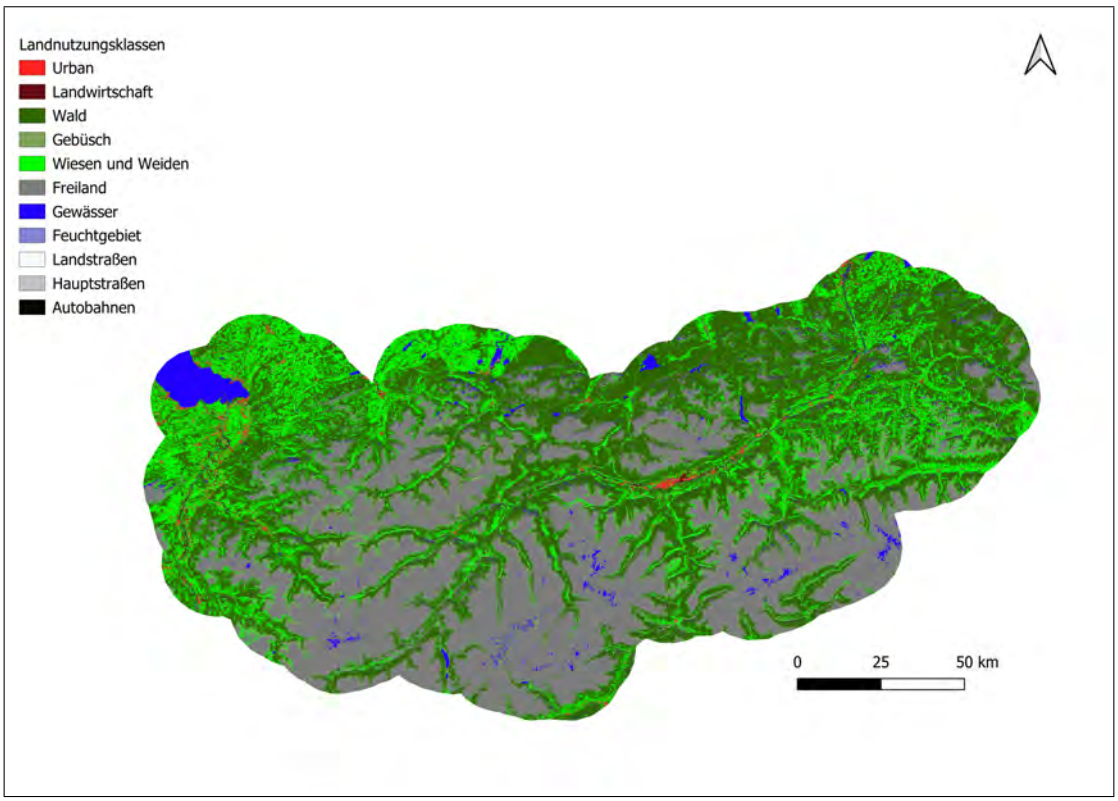


Abbildung 40: Vorarlberg und West Tirol.

5.3 Skripte zur Erkennung potentieller Wildkatzenkorridore

PotentialCorridorFinder.py

```
# ------------------------------------------------------------
# Potential Corridor Analysis Script
# ------------------------------------------------------------
# This script analyzes potential wildlife corridors between neighboring polygons
    in various Austrian states.
# It leverages spatial data processing techniques such as polygon clipping,
    pathfinding, and graph analysis.
# Author: Luka Kern
# Date: 12.08.2023

# Libraries Used:
# - os: Operating system-related operations and file handling.
# - geopandas: Managing geospatial data, spatial analysis, and manipulation.
# - networkx: Graph-based algorithms for pathfinding and analysis.
# - numpy: Numerical operations on arrays, particularly for raster data.
# - pandas: Data manipulation and analysis.
# - pyproj: Geospatial transformations and projections.
# - rasterio: Handling raster data, opening and manipulating raster datasets.
# - rtree: Efficient spatial indexing for nearest neighbor search.
# - scipy: Scientific computing libraries for various algorithms.
# - shapely.geometry: Creating and manipulating geometric shapes.
# - tqdm: Displaying progress bars for loops and tasks.
# - warnings: Handling warning messages.

# Functions:
# - clip_raster_by_extent: Clips a raster dataset by the extent of given
    polygons.
# - calculate_weight: Calculates weight based on heatmap and land use values.
# - find_neighbouring_polys: Finds neighboring polygons for each polygon in a
    GeoDataFrame.
# - find_vertex_pairs: Finds pairs of nearest vertices between polygons.
# - find_potential_corridors: Identifies potential corridors between pairs of
    points using land use and heatmap data.

# Main Code:
# - Iterates through Austrian states for data processing.
# - Loads polygons, heat map, and land use raster data.
# - Finds pairs of neighboring polygons that are close but don't touch.
# - Processes each pair of neighboring polygons:
#   - Finds vertex pairs for potential paths between polygons.
#   - Determines potential corridors using land use and heat map data.
# - Concatenates and saves the results as shapefiles for each state.
```

```
# Note: Ensure the appropriate paths and data sources are provided.
# ------------------------------------------------------------------


# Import required libraries
import os
import random
import warnings
import signal
import rtree

import geopandas as gpd
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import pandas as pd
import pyproj
import rasterio
from rasterio.windows import from_bounds
from rtree import index as rtree_index
from scipy.spatial.distance import euclidean
from scipy.spatial import KDTree
from scipy.spatial import cKDTree
from shapely.geometry import LineString, Point, Polygon, box
from tqdm import tqdm


# ================================================================================
# ================================================================================
# FUNCTIONS
# ================================================================================
# ================================================================================

# Define a custom exception for timeout
class TimeoutError(Exception):
    """Exception raised for timeout situations."""
    pass

# Define the timeout handler function
def timeout_handler(signum, frame):
    """Raise a TimeoutError when a timeout occurs."""
    raise TimeoutError("Timeout occurred while finding the shortest path.")

def deprecated_function():
    """Warn about deprecated usage."""
    warnings.warn("deprecated", DeprecationWarning)
```

```python
def clip_raster_by_extent(raster, gdf, buffer_distance=500, save_file=False, output_path=None):
    """
    Clip a raster by the extent of given polygons.

    Args:
        raster (rasterio.DatasetReader): The input raster dataset.
        gdf (geopandas.GeoDataFrame): The GeoDataFrame containing polygons.
        buffer_distance (float): Buffer distance around the extent (default is 500).
        save_file (bool): Whether to save the clipped raster as a file (default is False).
        output_path (str): Output path for the clipped raster file (default is None).

    Returns:
        tuple: A tuple containing the clipped raster data and its metadata.
    """
    xmin, ymin, xmax, ymax = gdf.total_bounds

    # Create a buffer around the extent
    bbox = box(xmin, ymin, xmax, ymax)
    buffered_bbox = bbox.buffer(buffer_distance)

    # Calculate the window from the buffered polygon bounds
    window = from_bounds(*buffered_bbox.bounds, transform=raster.transform)

    # Read the clipped raster data
    raster_clipped = raster.read(window=window)
    raster_transform = rasterio.windows.transform(window, raster.transform)

    # Update the metadata for the clipped raster
    raster_meta = raster.meta.copy()
    raster_meta.update({
        'transform': raster_transform,
        'width': window.width,
        'height': window.height,
        'bounds': (xmin, ymin, xmax, ymax)
    })

    # Save the clipped raster as a file if specified
    if save_file:
        if output_path is None:
            output_path = "clipped_raster.tif"
        with rasterio.open(output_path, "w", **raster_meta) as dst:
            dst.write(raster_clipped)
        print(f"Clipped raster saved as: {output_path}")
```

```
    return raster_clipped, raster_meta

def calculate_weight(heatmap_value, landuse_value, heatmap_nodata_value,
    landuse_nodata_value, heatmap_multiplier=3):
    """
    Calculate the weight based on heatmap and land use values.

    Args:
        heatmap_value (float): Heatmap value for the location.
        landuse_value (int): Land use value for the location.
        heatmap_nodata_value (float): NODATA value for heatmap.
        landuse_nodata_value (int): NODATA value for land use.
        heatmap_multiplier (float): Multiplier for heatmap-based weight (default
    is 3).

    Returns:
        float: Calculated weight.
    """
    # Handle NODATA values
    if heatmap_value == heatmap_nodata_value:
        heatmap_value = -0.99999
    if landuse_value == landuse_nodata_value:
        landuse_value = 0

    # Define a lookup dictionary for land-use weights
    landuse_weights = {
        0: 1000000,  # NODATA is most strictly avoided
        1: 1000,     # Urban area -- (1/1000)
        2: 500,      # Agriculture -- (1/500)
        3: 1,        # Forest is preferred -- (1)
        4: 500,      # Shrublands -- (1/500)
        5: 500,      # Grasslands -- (1/500)
        6: 1000,     # Barelands -- (1/1000)
        7: 1000,     # Waterbodies -- (1/1000)
        8: 1000,     # Wetlands -- (1/1000)
        9: 1000,     # Streets are avoided -- (1/1000)
        10: 1000,    # Streets are avoided -- (1/1000)
        11: 1000     # Streets are avoided -- (1/1000)
    }

    # Calculate the final weight based on the heatmap and land use values
    weight = 1 / (heatmap_value + 1) * heatmap_multiplier
    if landuse_value in landuse_weights:
        weight *= landuse_weights[landuse_value]

    return weight
```

```
def find_neighbouring_polys(gdf):
    """
    Find neighbouring polygons for each polygon in the GeoDataFrame.

    Args:
        gdf (geopandas.GeoDataFrame): Input GeoDataFrame with polygons and "fid"
    attribute.

    Returns:
        list: List of pairs of neighbouring polygon "fid"s.
    """
    # Create a spatial index for efficient nearest neighbor search
    spatial_index = rtree.index.Index()
    for row_index, row in gdf.iterrows():
        geometry = row['geometry']
        spatial_index.insert(row_index, geometry.bounds)

    neighbours_set = set()

    # Set up the progress bar for better user feedback
    pbar = tqdm(total=len(gdf), desc="Finding neighbours")

    for index, row in gdf.iterrows():
        polygon = row['geometry']
        fid = row['fid']
        centroid = polygon.centroid

        for direction in ['north', 'east', 'south', 'west']:
            nearest_distance = float('inf')  # Initialize with a large distance
            nearest_neighbour_fid = None

            # Determine search area based on direction
            if direction == 'north':
                candidates = list(spatial_index.intersection((centroid.x,
    centroid.y, centroid.x, float('inf'))))
            elif direction == 'east':
                candidates = list(spatial_index.intersection((centroid.x,
    centroid.y, float('inf'), centroid.y)))
            elif direction == 'south':
                candidates = list(spatial_index.intersection((centroid.x, float(
    '-inf'), centroid.x, centroid.y)))
            elif direction == 'west':
                candidates = list(spatial_index.intersection((float('-inf'),
    centroid.y, centroid.x, centroid.y)))

            # Find the nearest neighboring polygon in the given direction
```

```
            for candidate in candidates:
                if candidate != index:
                    neighbour_polygon = gdf.loc[candidate, 'geometry']
                    neighbour_centroid = neighbour_polygon.centroid

                    distance = centroid.distance(neighbour_centroid)
                    if distance < nearest_distance:
                        nearest_distance = distance
                        nearest_neighbour_fid = gdf.loc[candidate, 'fid']

            # Add the pair to the set if a valid neighbor is found
            if nearest_neighbour_fid:
                pair = tuple(sorted((fid, nearest_neighbour_fid)))
                neighbours_set.add(pair)

        # Update the progress bar to reflect processing status
        pbar.update(1)

    # Close the progress bar upon completion
    pbar.close()

    neighbours_list = list(neighbours_set)

    return neighbours_list

def find_vertex_pairs(gdf, n_pairs=10, save_file=False, output_path=None):
    """
    Find pairs of nearest vertices between two polygons in a GeoDataFrame.

    Args:
        gdf (geopandas.GeoDataFrame): Input GeoDataFrame with a pair of polygons
    .
        n_pairs (int): Number of vertex pairs to find (default is 10).
        save_file (bool): Whether to save the resulting points as a shapefile (
    default is False).
        output_path (str): Output path for the shapefile (default is None).

    Returns:
        geopandas.GeoDataFrame: GeoDataFrame containing the pairs of points.
    """
    # Example polygons from the input GeoDataFrame
    polygon1 = gdf.geometry.iloc[0]
    polygon2 = gdf.geometry.iloc[1]

    # Extract vertices from the polygons
    vertices1 = list(polygon1.exterior.coords)
    vertices2 = list(polygon2.exterior.coords)
```

```
    # Build a spatial index for the vertices of polygon2
    tree = cKDTree(vertices2)

    # Find the nearest neighbor in polygon2 for each vertex in polygon1
    distances, indices = tree.query(vertices1)

    # Sort the distances and get the indices of the closest vertex pairs
    closest_indices = np.argsort(distances)

    # Create a set to keep track of added points
    added_points = set()

    # Retrieve the closest vertex pairs without repeating points
    closest_pairs = []
    for i in closest_indices:
        point1 = Point(vertices1[i])
        point2 = Point(vertices2[indices[i]])

        # Check if both points are already added
        if point1 in added_points or point2 in added_points:
            continue

        closest_pairs.append((point1, point2))
        added_points.add(point1)
        added_points.add(point2)

        # Stop if the desired number of pairs is reached
        if n_pairs is not None and len(closest_pairs) == n_pairs:
            break

    # Create a GeoDataFrame to store the points
    geometry = [pair[0] for pair in closest_pairs] + [pair[1] for pair in
closest_pairs]
    pairing = list(range(1, len(closest_pairs) + 1)) + list(range(1, len(
closest_pairs) + 1))  # Assign unique identifiers for each pair
    points_df = gpd.GeoDataFrame({"pairing": pairing, "geometry": geometry}, crs
=gdf.crs)

    # Save the GeoDataFrame as a shapefile if specified
    if save_file:
        if output_path is None:
            output_path = "points_of_interest.shp"
        points_df.to_file(output_path)
        print(f"Points of interest saved as: {output_path}")

    return points_df
```

```
def find_potential_corridors(point_pairs, landuse, heatmap, heatmap_nodata_value
    , landuse_nodata_value, timeout_duration=6000, max_node_capacity=6000000):
    """
    Find potential corridors between pairs of points.

    Args:
        point_pairs (GeoDataFrame): Vertex pairs on the edge of a polygon pair.
        landuse (numpy.ndarray): Landuse raster.
        heatmap (numpy.ndarray): Heatmap of wildcat visits.
        heatmap_nodata_value (float): NODATA value of heatmap.
        landuse_nodata_value (float): NODATA value of landuse.
        timeout_duration (int, optional): Timeout in seconds for path finding.
        max_node_capacity (int, optional): Maximum node capacity for graph.

    Returns:
        gdf_line_paths (GeoDataFrame): GeoDataFrame of potential corridor paths.
    """

    # Clip heatmap and landuse rasters to match point_pairs extent
    heatmap_clipped, heatmap_clipped_meta = clip_raster_by_extent(heatmap,
    point_pairs)
    landuse_clipped, landuse_clipped_meta = clip_raster_by_extent(landuse,
    point_pairs)

    # Create a graph for pathfinding
    G = nx.Graph()

    # Extract the first band of the clipped arrays
    heatmap_clipped = heatmap_clipped[0]
    landuse_clipped = landuse_clipped[0]


# Iterate over the clipped pixels to build the graph
    with tqdm(total=heatmap_clipped.shape[0] * heatmap_clipped.shape[1], desc='
    Building graph') as pbar:
        for (row, col), heatmap_value in np.ndenumerate(heatmap_clipped):
            if not np.isnan(heatmap_value):
                landuse_value = landuse_clipped[row, col]

                # Calculate centroid coordinates
                centroid = landuse_clipped_meta["transform"] * (col + 0.5, row +
    0.5)
                centroid_x, centroid_y = centroid[0], centroid[1]

                # Add the node to the graph
                node = (centroid_x, centroid_y)
```

```
            G.add_node(node)

            # Connect the node to its neighboring nodes
            neighbours = [(row, col - 1), (row - 1, col), (row, col + 1), (
row + 1, col)]
            for neighbour_row, neighbour_col in neighbours:
                if 0 <= neighbour_row < heatmap_clipped.shape[0] and 0 <=
neighbour_col < heatmap_clipped.shape[1]:
                    neighbour_heatmap_value = heatmap_clipped[neighbour_row,
 neighbour_col]
                    neighbour_landuse_value = landuse_clipped[neighbour_row,
 neighbour_col]

                    if not np.isnan(neighbour_heatmap_value):
                        # Calculate neighbor's centroid coordinates
                        neighbour_centroid = landuse_clipped_meta["transform
"] * (neighbour_col + 0.5, neighbour_row + 0.5)
                        neighbour_centroid_x, neighbour_centroid_y =
neighbour_centroid[0], neighbour_centroid[1]

                        # Calculate weight and add edge
                        neighbour_weight = calculate_weight(
neighbour_heatmap_value, neighbour_landuse_value, heatmap_nodata_value,
landuse_nodata_value)
                        G.add_edge(node, (neighbour_centroid_x,
neighbour_centroid_y), weight=neighbour_weight)

        pbar.update(1)


if len(G.nodes) < max_node_capacity:
    print("Locating start and end locations for potential paths")
    line_paths = []

    # Create a KDTree from the graph nodes
    nodes = [node for node in G.nodes]
    kdtree = KDTree(nodes)

    # Group the point pairs by the "pairing" column
    grouped_pairs = point_pairs.groupby('pairing')

    for i, group in grouped_pairs:
        point1 = group.geometry.iloc[0]
        point2 = group.geometry.iloc[1]

        # Find the index of the closest nodes
        closest_node_index1 = kdtree.query((point1.x, point1.y))[1]
```

```
            closest_node_index2 = kdtree.query((point2.x, point2.y))[1]

            # Get the closest node coordinates
            closest_node_point1 = nodes[closest_node_index1]
            closest_node_point2 = nodes[closest_node_index2]

            start_loc = closest_node_point1
            end_loc = closest_node_point2

            # Check if source and target nodes exist in the graph
            if start_loc in G.nodes and end_loc in G.nodes:
                try:
                    print("Finding the shortest path between start_loc and
end_loc")
                    # Apply the timeout to the code block
                    signal.signal(signal.SIGALRM, timeout_handler)
                    signal.alarm(timeout_duration)

                    # Find the shortest path between the start_loc and end_loc
                    shortest_path = nx.astar_path(G, source=start_loc, target=
end_loc, weight='weight')

                    if len(shortest_path) > 1:
                        # Create a LineString object from the coordinates
                        line_path = LineString(shortest_path)
                        line_paths.append(line_path)
                        print(f"Path {i} of {grouped_pairs.ngroups} appended
successfully.")
                    else:
                        print(f"No valid path found between start_loc and
end_loc.")
                except nx.NodeNotFound as e:
                    print("Error: Either source or target is not in the graph.")
                except TimeoutError as e:
                    print(str(e))
                finally:
                    # Reset the alarm
                    signal.alarm(0)
            else:
                print("Error: Either source or target is not within the masked
area of interest.")


        if not line_paths:
            warnings.warn("No corridors found for this polygon pair.")
            gdf_line_paths = None
        else:
```

```
            # Convert LineString objects to Pandas Series
            line_series = [pd.Series([line], name='geometry') for line in
    line_paths]

            # Concatenate the series into a single DataFrame
            concatenated_df = pd.concat(line_series, ignore_index=True).
    reset_index(drop=True)

            # Convert the concatenated DataFrame to a GeoDataFrame
            gdf_line_paths = gpd.GeoDataFrame(concatenated_df, geometry='
    geometry')

            # Set crs
            gdf_line_paths.crs = point_pairs.crs

    else:
        gdf_line_paths = None

    return gdf_line_paths


# ============================================================================
# ============================================================================
# Main code
# ============================================================================
# ============================================================================


# List of Austrian states for processing
austria_states = ["Salzburg"] # "Kaernten", "Steiermark", "Burgenland", "
    Oberoesterreich", "Tirol", "Vorarlberg", "Niederoesterreich"]

# Iterate through each state for data processing
for state in austria_states:
    # Set working directory to the current state's data folder
    os.chdir(f"/Users/lukakern/Desktop/wildcats/dat/{state}")

    # Load polygons from shapefile for the current state
    all_polygons = gpd.read_file(f"shp/{state}Patches.shp")

    # Assign a unique 'fid' to each polygon
    all_polygons['fid'] = range(1, len(all_polygons) + 1)

    # Load heat map raster for the current state
    heatmap_path = f"rst/{state}Heatmap.tif"
    heatmap_src = rasterio.open(heatmap_path)
    heatmap_nodata_value = heatmap_src.nodatavals[0]
```

```
    # Load land use raster for the current state
    landuse_path = f"rst/{state}Landuse.tif"
    landuse_src = rasterio.open(landuse_path)
    landuse_nodata_value = landuse_src.nodatavals[0]

    # Create an output directory for saving results
    output_dir = "output"
    os.makedirs(output_dir, exist_ok=True)

    # Suppress deprecated function warnings
    with warnings.catch_warnings():
        warnings.simplefilter("ignore")
        deprecated_function()  # Call deprecated function

    # Find pairs of neighboring polygons that are close but don't touch
    neighbouring_polys_fids = find_neighbouring_polys(all_polygons)

    # Initialize lists to store line paths and points
    all_line_paths_lst = []
    all_points_lst = []

    # Process each pair of neighboring polygons
    for i, fid_pair in enumerate(neighbouring_polys_fids):
        print(f"Processing neighboring polygons pair {i+1}/{len(
neighbouring_polys_fids)} in {state}.")

        fid1, fid2 = fid_pair
        # Select the current fid pair's rows from the polygons DataFrame
        selected_polys = all_polygons[(all_polygons['fid'] == fid1) | (
all_polygons['fid'] == fid2)]

        # Find vertex pairs for potential paths between polygons
        point_pairs = find_vertex_pairs(selected_polys, n_pairs=20)

        # Store the point pairs list
        all_points_lst.append(point_pairs)

        # Find potential corridors using land use and heat map data
        line_paths = find_potential_corridors(point_pairs, landuse_src,
heatmap_src, heatmap_nodata_value, landuse_nodata_value)

        # Store the line paths list
        all_line_paths_lst.append(line_paths)

    # Concatenate all line path and point DataFrames
    paths_concat_df = pd.concat(all_line_paths_lst, ignore_index=True).
```

```
    reset_index(drop=True)
    points_concat_df = pd.concat(all_points_lst, ignore_index=True).reset_index(
    drop=True)

    # Convert concatenated DataFrames to GeoDataFrames
    gdf_all_line_paths = gpd.GeoDataFrame(paths_concat_df, geometry='geometry')
    gdf_all_points = gpd.GeoDataFrame(points_concat_df, geometry='geometry')

    # Set the CRS for the GeoDataFrames
    gdf_all_line_paths.crs = all_polygons.crs
    gdf_all_points.crs = all_polygons.crs

    # Save the results as shapefiles
    gdf_all_line_paths.to_file(f"output/{state}_potential_corridors.shp")
    gdf_all_points.to_file(f"output/{state}_start_end_points.shp")
```

HeatmapLanduseProcessing.py

```
# ------------------------------------------------------------
# Wildlife Corridor Analysis Script
# ------------------------------------------------------------
# This script preprocesses raster data to analyze potential wildlife corridors
    in Austrian states. It performs
# operations such as raster alignment, heatmap to polygon conversion, and
    landuse to polygon conversion.
# Author: Luka Kern
# Date: 12.08.2023
# ------------------------------------------------------------



# Libaries Used:
# - rasterio: Handling raster data, opening and manipulating raster datasets.
# - rasterio.features: Working with features extracted from raster datasets.
# - geopandas: Managing geospatial data, spatial analysis, and manipulation.
# - numpy: Numerical operations on arrays, particularly for raster data.
# - os: Directory operations, managing file paths.
# - tqdm: Displaying progress bars for loops and tasks.
# - shapely.geometry: Creating and manipulating geometric shapes.
# - rasterio.transform: Handling raster transformation operations.

# States Analyzed: The script focuses on Austrian states, by using a provided
    list

# Processing Steps:

#    1. Import Libraries: Import necessary Python libraries for the analysis.
#    2. Define Parameters: Set up parameters, including the list of Austrian
```

```
    states to analyze and processing flags for specific steps.
#     3. Main Loop: For each state in the list of Austrian states:
#           Set Working Directory: Change the working directory to the state's
    data directory.
#           Open Raster Data: Open two raster files: one for a heatmap and
    another for land use.
#           Optional: Align Raster: If specified, align the land use raster with
     the heatmap raster through reprojection and resampling.
#           Optional: Heatmap to Polygons: If specified, convert heatmap raster
    cells above a threshold into polygons, dissolve them, and save as a shapefile
    .
#           Optional: Landuse to Polygons: If specified, convert land use raster
     cells representing forests into binary raster cells, then into forest
    polygons, and save as a shapefile.
#     4. Completion Message: Print a message indicating that the corridor
    analysis has been completed for all states.


# Import required libraries
import rasterio
from rasterio.features import shapes
import geopandas as gpd
import numpy as np
import os
from tqdm import tqdm
from shapely.geometry import shape

# Define Austrian states to process
austria_states = ["Salzburg"] # "Kaernten", "Steiermark", "Burgenland", "
    Oberoesterreich", "TirolVorarlberg", "Niederoesterreich"]

# Define processing flags
align_landuse_raster = False
heatmap_to_polys = False
landuse_to_polys = True

# Loop through each state for analysis
for state in austria_states:
    print(f"Processing {state}...")

    # Set working directory
    os.chdir(f"/Users/lukakern/Desktop/wildcats/dat/{state}")

    # Open the raster data
    with rasterio.open(f"rst/{state}Heatmap.tif") as heatmap_rst:
        with rasterio.open(f"rst/{state}Landuse.tif") as landuse_rst:
```

```
        # ALIGN RASTER
        if align_landuse_raster:
            # Define transformation parameters
            transform_params = rasterio.transform.from_bounds(
                *heatmap_rst.bounds, width=landuse_rst.width, height=
landuse_rst.height
            )

            # Create a transformed landuse raster
            transformed_landuse_rst = rasterio.open(
                f"rst/{state}TransformedLanduse.tif",
                "w",
                driver="GTiff",
                height=landuse_rst.height,
                width=landuse_rst.width,
                count=landuse_rst.count,
                dtype=landuse_rst.dtypes[0],
                crs=heatmap_rst.crs,
                transform=transform_params
            )

            # Reproject the landuse raster
            rasterio.warp.reproject(
                source=rasterio.band(landuse_rst, 1),
                destination=rasterio.band(transformed_landuse_rst, 1),
                src_transform=landuse_rst.transform,
                src_crs=landuse_rst.crs,
                dst_transform=transform_params,
                dst_crs=heatmap_rst.crs,
                resampling=rasterio.enums.Resampling.nearest
            )

            transformed_landuse_rst.close()

        # HEATMAP TO POLYGONS
        if heatmap_to_polys:
            # Set the threshold value for the heatmap
            heatmap_threshold = 1

            # Convert the data type of the heatmap raster to a supported
type
            heatmap_data = heatmap_rst.read(1).astype(np.float32)

            # Initialize a list to store heatmap polygons
            heatmap_polygons = []

            # Convert raster cells above threshold to polygons
```

```
            for geometry, value in tqdm(features.shapes(heatmap_data,
transform=heatmap_rst.transform), desc="Processing heatmap"):
                if value >= heatmap_threshold:
                    heatmap_polygons.append(shape(geometry))

            # Convert the list of polygons to a GeoDataFrame
            heatmap_polygons = gpd.GeoDataFrame(geometry=heatmap_polygons)

            # Dissolve polygons based on their values
            dissolved_polygons = heatmap_polygons.dissolve(by=
heatmap_polygons.geometry.values)

            # Assign CRS to the dissolved polygons
            dissolved_polygons.crs = heatmap_rst.crs

            # Save the dissolved polygons as a shapefile
            dissolved_polygons.to_file(f"shp/{state}HeatmapPolygons.shp")

        # LANDUSE TO POLYGONS
        if landuse_to_polys:
            # Use the transformed_landuse_rst if available, otherwise use
the original
            if not align_landuse_raster:
                transformed_landuse_rst = landuse_rst

            # Set the value for the landuse code representing forests
            landuse_forest_code = 3

            # Create binary raster based on the threshold and landuse code
            landuse_binary = transformed_landuse_rst.read(1) ==
landuse_forest_code

            # Convert the data type of landuse_binary  to a supported type
            landuse_binary = landuse_binary.astype(np.uint8)

            # Initialize a list to store landuse polygons
            landuse_polygons = []

            # Convert binary raster cells to polygons
            for geometry, value in tqdm(shapes(landuse_binary, transform=
heatmap_rst.transform), desc=f"Processing landuse in {state}"):
                if value == 1:
                    landuse_polygons.append(shape(geometry))

            # Convert the GeoSeries to have a common index type
            landuse_polygons = gpd.GeoSeries(landuse_polygons).reset_index(
drop=True)
```

```
                # Assign CRS to landuse_polygons
                landuse_polygons.crs = heatmap_rst.crs

                # Save landuse_polygons as shapefile
                landuse_polygons.to_file(f"shp/{state}ForestPolygons.shp")

print("Corridor analysis completed for all states.")
```

CorridorAreaCalculation.py

```
# ------------------------------------------------------------
# Potential Wildlife Corridor Analysis Script
# ------------------------------------------------------------
# This script processes potential wildlife corridors in Austrian states,
  identifying forested and non-forested areas,
# calculating their areas, and saving the results as shapefiles and a text file.
# Author: Luka Kern
# Date: 12.08.2023
# ------------------------------------------------------------

# Libraries Used:
# - os for directory operations
# - geopandas for geospatial data processing
# - pandas for data manipulation

# States Analyzed: The script focuses on Austrian states, by using a provided
  list

# Processing Steps:
# 1. Loads potential corridor lines, patches, and forest polygons.
# 2. Buffers corridor lines by 50 meters and converts to a GeoDataFrame.
# 3. Clips buffered lines by patch polygons to identify potential corridors.
# 4. Determines forested areas by overlaying forest land use and corridors.
# 5. Corrects geometry validity issues if present.
# 6. Computes non-forested areas by calculating differences.
# 7. Creates GeoDataFrames for forested and non-forested areas.
# 8. Adds an attribute to indicate the presence of forest.
# 9. Saves forested and non-forested areas as separate shapefiles.
# 10. Combines GeoDataFrames, calculates and displays areas.
# 11. Writes corridor area statistics to a text file.
# 12. Saves the combined corridor areas as a shapefile.

# ------------------------------------------------------------



# Import required libraries
```

```
import os
import geopandas as gpd
import pandas as pd

# Define Austrian states to process
austria_states = ["Salzburg"] # "Kaernten", "Steiermark", "Burgenland", "
    Oberoesterreich", "TirolVorarlberg", "Niederoesterreich"]

# Loop through each state for corridor analysis
for state in austria_states:
    print(f"Working on corridors of {state}.")

    # Change working directory to the state's data folder
    os.chdir(f"/Users/lukakern/Desktop/wildcats/dat/{state}")

    # Load data: potential corridor lines, patches, and forest polygons
    corridor_lines = gpd.read_file(f"output/{state}_potential_corridors.shp")
    patches = gpd.read_file(f'shp/{state}Patches.shp')
    forest_landuse = gpd.read_file(f'shp/{state}ForestPolygons.shp')

    # Buffer the corridor lines by 50 meters
    print("Buffering the corridor lines ...")
    buffered_corridor_lines = corridor_lines.buffer(50)

    # Convert the buffered lines GeoSeries to a GeoDataFrame
    buffered_corridor_lines = gpd.GeoDataFrame(geometry=buffered_corridor_lines)

    # Clip the buffered lines by the patches polygons (calculate the difference)
    corridors = buffered_corridor_lines

    # Identify forested areas
    print("Checking which areas are forested ...")
    corridor_forest = gpd.overlay(forest_landuse, corridors, how='intersection')

    # Check and correct geometry validity
    if not corridors.geometry.is_valid.all():
        print("Corrected corridor geometry!")
        corridors.geometry = corridors.geometry.buffer(0)
    if not corridor_forest.geometry.is_valid.all():
        print("Corrected forested corridor geometry!")
        corridor_forest.geometry = corridor_forest.geometry.buffer(0)

    # Calculate non-forested areas
    corridor_non_forest = corridors.geometry.difference(corridor_forest.
    unary_union)

    # Create GeoDataFrame for non-forested corridor areas
```

```
    corridor_non_forest_gdf = gpd.GeoDataFrame(geometry=corridor_non_forest)

    # Add attribute indicating forest presence
    corridor_forest['is_forest'] = 1
    corridor_non_forest_gdf['is_forest'] = 0

    # Save forested and non-forested areas as shapefiles
    corridor_forest.to_file(f'output/{state}_corridor_forest.shp')
    corridor_non_forest_gdf.to_file(f'output/{state}_corridor_non_forest.shp')

    # Concatenate forested and non-forested GeoDataFrames
    corridor_combined = pd.concat([corridor_forest, corridor_non_forest_gdf])

    # Calculate and display areas
    print("Calculating the areas ...")
    corridor_combined['area'] = corridor_combined.geometry.area
    overall_corridor_area = corridor_combined.geometry.area.sum()
    is_forest_area = corridor_combined[corridor_combined['is_forest'] == 1].
   geometry.area.sum()
    is_not_forest_area = corridor_combined[corridor_combined['is_forest'] == 0].
   geometry.area.sum()

    # Write areas to a text file
    print("Writing the areas to a text file")
    with open('output/Corridor_area.txt', 'a') as file:
        file.write("Overall potential corridor area:\n" + str(
   overall_corridor_area) + '\n')
        file.write("Forested area:\n" + str(is_forest_area) + '\n')
        file.write("Not forested area:\n" + str(is_not_forest_area) + '\n')

    print('Areas written to file.')

    # Save combined corridor areas as shapefile
    print("Saving the corridors as shape file ...")
    corridor_combined.to_file(f'output/{state}_potential_corridor_areas.shp')

print("Corridor analysis completed for all states.")
```