Hochschule München

Fakultät für Maschinenbau, Fahrzeugtechnik, Flugzeugtechnik

Bachelorarbeit

Entwicklung einer UAV-Steuerungssoftware zur optimierten autonomen Rehkitzdetektion

Patrick Berghäuser

zur Erlangung des akademischen Grades
Bachelor of Science

Studiengang:	Luft- und Raumfahrttechnik
Eingereicht am:	27. September 2020
Betreuer:	Prof. Dr. Frank Palme
Institutsbetreuer:	Dr. Martin Israel

Selbstständigkeitserklärung

Erklärung zur Bachelorarbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

München den 27. September 2020

Zusammenfassung

Da jedes Jahr um die 100.000 Rehkitze in Deutschland bei Mäharbeiten verenden, wurde sich in dieser Arbeit mit dem Aufspüren von Jungwild in Wiesen beschäftigt. Zwar gibt es bereits Methoden für die Rettung von Rehkitzen, jedoch sind viele dieser zu teuer, zu personalaufwendig oder zu zeitintensiv für den breiten Einsatz. Ziel dieser Arbeit war es, eine Steuerungssoftware für ein UAV zu entwickeln, welche autonom Wildtiere entdeckt, anfliegt und verifiziert.

Ermöglicht wurde dies durch ein Zusammenspiel einer Infrarotkamera, eines Einplatinencomputers sowie einem mit Sensorsystemen ausgestatteten Flugreglers. Durch die unmittelbare Bildverarbeitung mittels neuronalem Netz konnte während des Fluges direkt auf detektierte Objekte in den Infrarotbildern reagiert werden. Mittels Georeferenzierung wurden aus den Bilddaten und unter Verwendung der Sensordaten des Flugreglers die GPS-Positionen der erkannten Objekte bestimmt. Für eine genauere Betrachtung der Objekte wurde die auf dem Flugregler hinterlegte Mission auf den Einplatinencomputer übertragen und soweit manipuliert, dass das Trägersystem die berechneten Koordinaten mit geringerer Flughöhe anfliegt. Durch die geschaffene verringerte Distanz zum Objekt konnte in den Bildern ein Informationsgewinn um Faktor 8 gegenüber Bildern der normalen Flughöhe geschaffen werden. Hierdurch ergeben sich neue Möglichkeiten für eine spätere genauere Verifikation oder Verarbeitung. Zusätzlich wurde ein Clusteringalgorithmus implementiert, welcher verhindert, dass gleiche Objekte mehrmals im Laufe einer Mission angeflogen werden.

Durch die lokale Speicherung der während des Fluges ermittelten Fundstellenkoordinaten kann nach der Landung des Systems eine Rettung der gefundenen Jungtiere stattfinden. Auch eine automatisierte Abspeicherung der Infrarotbilder wurde implementiert, sodass die Fundstellen anhand des Bildmaterials manuell überprüft werden können. Es wurde somit ein Grundstein für die vollautonome Rehkitzsuche gelegt, welcher durch verbesserte neuronale Netze weiter ausgebaut werden könnte. Unter gewissen Umgebungsbedingungen wurde die Flächenleistung der automatisierten Rehkitzsuche auf 19 ha/h berechnet, wodurch das System schon jetzt unter gegebenen Konditionen eine höhere Flächenleistung gegenüber etablierten Verfahren hat. Zusätzlich kann durch die autonome Erkennung und Verifikation bei der Rehkitzrettung Zeit und Personal gespart werden.

Summary

Since around 100,000 fawns die every year in Germany while mowing, this work focused on tracking down young deers in meadows. Although there are already methods for rescuing fawns, many of them are too expensive, labor-intensive or time-consuming for widespread use. The aim of this work was to develop a control software for a UAV, which autonomously discovers, flies to and verifies wild fawns.

This was made possible by the interaction of an infrared camera, a single-board computer and a flight controller equipped with additional sensor systems. Immediate image processing by a neural network made it possible to react directly to objects detected in the infrared images during the flight. By means of georeferencing, the GPS-positions of the detected objects were determined from the image data and with help by the sensor data of the flight controller. For a closer look at the objects, the mission stored on the flight controller was transferred to the single-board computer and manipulated to the extent that the carrier system flies to the calculated coordinates at a lower altitude. Due to the created reduced distance to the object, information in the images by a factor of 8 compared to images of normal flight altitude was gained. This results in new possibilities for later more precise verification or processing. In addition, a clustering algorithm was implemented which prevents the same objects from being approached several times in the course of one mission.

The on-board storage of the location coordinates determined during the flight makes it possible that the found animals can be rescued after the system has landed. Automated storage of the infrared images has also been implemented so that the locations can be examined even manually using the image material. Accordingly a solid foundation for the fully autonomous fawn search was laid, which could be further expanded by improved neural networks. Under certain environmental conditions, the area performance of the automated fawn search was calculated to be 19 ha/h, whereby the system already has a higher area performance compared to established methods under given conditions. In addition, the autonomous detection and verification of the fawn rescue can save time and personnel.

Inhaltsverzeichnis

Symbolverzeichnis

Symbol	Einheit	Beschreibung
a_j		Aktivierungszustand eines Neurons
B	px	Breite der Bildebene
c	m	Kammerkonstante
c_x	px	x-Koordinate des Zentralpunktes in der Bildebene
c_y	px	y-Koordinate des Zentralpunktes in der Bildebene
d	m	Distanz zweier GPS-Koordinaten
h	m	Höhe des UAVs über Grund
H	px	Höhe der Bildebene
i		Patchnummer
k_1		Erster Koeffizient der radialen Verzeichnung
k_2		Zweiter Koeffizient der radialen Verzeichnung
k_3		Dritter Koeffizient der radialen Verzeichnung
n		Anzahl
net_j		Ausgabewert einer Propagierungsfunktion
O		Projektionszentrum
p_1		Erster Koeffizient der tangentialen Verzeichnung
p_2		Zweiter Koeffizient der tangentialen Verzeichnung
P_B	px	Breite eines Patches
P_H	px	Höhe eines Patches
R	m	Erdradius
s_{xy}		Schiefsymmetrische Koeffizient zwischen der x- und y-Achse
S	m	Schwadbreite
S_m	m	Mittlerer Abstand
t_1	s	Zeit bei Methode 1
t_2	s	Zeit bei Methode 2
w_{ij}		Gewichtung eines Eingabewertes zum Neuron
x	px	x-Koordinate in der Bildebene
x^*	px	x-Koordinate innerhalb eines Patches
x_i		Eingabewert Neuron

Symbol	Einheit	Beschreibung
X	m	X-Koordinate eines 3-D Punktes
ΔX_C	m	Versatz zwischen Projektionszentrum und UAV-Koordinatensystem
X_0	m	X-Koordinate des Projektionszentrums
y	px	y-Koordinate in der Bildebene
y^*	px	y-Koordinate innerhalb eines Patches
Y	m	Y-Koordinate eines 3-D Punktes
Y_0	m	Y-Koordinate des Projektionszentrums
Z	m	Z-Koordinate eines 3-D Punktes
Z_0	m	Z-Koordinate des Projektionszentrums
γ	°	Bahnwinkel des UAVs gegenüber Horizont
θ_j		Ausgabewert einer Propagierungsfunktion
κ	°	Hängewinkel der Projektionsebene
λ		Abbildungsmaßstab
μ	°	Rollwinkel des UAVs
ϕ	°	Nickwinkel der Projektionsebene
Φ	°	Breitengrad
χ	°	Azimut des UAVs
Ψ	°	Längengrad
ω	°	Azimuth der Projektionsebene

Abkürzungsverzeichnis

CNN	Convolutional Neural Network
FOV	Field of View
GPS	Global Positioning System
INS	Inertial Navigation System
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
POI	Point of Interest
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
UTM	Universal Transverse Mercator

1 Einleitung

1.1 Einsatzgebiete unbemannter Luftfahrzeuge

Durch die steigende Rechenleistung und die immer günstiger werdenden Elektronikkomponenten haben unbemannte Luftfahrzeuge in den letzten Jahren immer mehr an Nachfrage gewonnen. Großen Anklang findet die Nutzung von unbemannten Luftfahrzeugen in Bereichen, in welchen die Beobachtung durch Satelliten zu ungenau ist bzw. der Einsatz von bemannten Luftfahrzeugen zu teuer, oder aufgrund der Größe schlicht nicht möglich ist. Zu diesen Bereichen gehören beispielsweise die Topografie [Nex14, Man13] oder die Inspektion und Analyse von Gebäuden [Nex19, Che19].

Erst in letzter Zeit haben sich durch die Kombination von UAVs und die Verarbeitung deren Aufnahmen durch künstliche Intelligenz neue Einsatzgebiete erschlossen. Einsatzgebiete wie die automatisierte Wildtierzählung [Kel17], oder der Hindernisvermeidung [Wan20].

1.2 Motivation und Verfahren zur Jungwildrettung

Laut der Tierschutzorganisation *PeTA* sterben deutschlandweit jährlich etwa 100.000 Rehe durch Mäharbeiten [Gol20]. Häufig handelt es sich bei den verunglückten Tieren um Rehkitze, also die Jungtiere. Ein Vertreiben oder Aufspüren der Kitze ist selten problemlos. Zum einen haben die Jungtiere einen sogenannten Drückinstinkt, welcher die Tiere daran hindert, bei drohender Gefahr von sich aus die Flucht zu ergreifen. Zum anderen sind die Tiere meist tief im hohen Gras versteckt und somit für Helfer kaum sichtbar. Beispielsweise ist eine Aufspürung mit Hunden durch den hohen Personalaufwand und die geringe Flächenleistung nicht für jeden Landwirt wirtschaftlich tragbar.

Um die Flächenleistung, also die abgesuchte Fläche pro Zeit, zu erhöhen werden zunehmend mit Wärmebildkameras ausgestattete UAV-Systeme eingesetzt. Bisher gibt es für die Rehkitzsuche zwei etablierte Verfahren: Live View und Post Processing.

Beim Live View Verfahren werden drei bis fünf Personen benötigt. Hierbei steuert eine Person das UAV, während eine weitere Person auf einem Bildschirm Wärmebildsignaturen im überflogenen Bereich ausmacht. Bei einer Sichtung auf dem Bildschirm werden die Läufer über Funk zu der Fundstelle gelotst, während das UAV über dem Objekt verweilt. Zum hohen Personalaufwand kommt negativ

die begrenzte Einsatzdauer hinzu. Das Verfahren kann nämlich nur in den frühen Morgenstunden zuverlässig eingesetzt werden, da die Wärmesignaturen bei zunehmender Sonneneinstrahlung schwerer erkennbar werden. Insgesamt kann mit dem Live View Verfahren eine Flächenleistung von maximal 10 ha/h erreicht werden.

Bei der Alternative, dem Post Processing Verfahren werden die Wärmesignaturen nicht im Flug erkannt, sondern nach der Landung im Bildmaterial. Während des Erkundens werden in regelmäßigen Abständen Bilder getätigt, welche dann am Boden manuell oder durch einen Algorithmus analysiert werden. Durch direkte Georeferenzierung werden dann anhand der zuvor aufgenommenen Bilder die Koordinaten der Fundorte bestimmt. Diese Koordinaten können an das GPS-Gerät eines Läufers übertragen werden. Während die Läufer die gefundenen Kitze auflesen, kann das System bereits das nächste Gebiet überfliegen. Somit kann insgesamt eine erhöhte Flächenleistung von etwa 30 ha/h erreicht werden. Problematisch bei dem Verfahren ist, dass ein in den Bildern erkanntes Objekt nicht ohne höheren Aufwand genauer untersucht werden kann. Somit ist es möglich, dass es sich bei dem Objekt gar nicht um ein Kitz handelt und der Läufer den Weg umsonst zurückgelegt hat. Zudem wird eine Person benötigt, welche die Auswertung der Bilder an einem Computer vornimmt [Zab20].

1.3 Aufgabenstellung

Die Arbeit ist aus einer Kooperation der Hochschule München und dem Deutschen Zentrum für Luft- und Raumfahrt hervorgegangen. Ziel dieser Arbeit ist die Entwicklung eines optimierten Verfahrens, um die Vorteile des Live View und Post Processing Verfahrens gewinnbringend zu kombinieren. Hierbei soll die begrenzte Flächenleistung des Live View Verfahrens durch automatisierte Erkennung und Verarbeitung der Rehkitzfundstellen kompensiert werden. Die automatisierte Erkennung soll eine auf dem mitfliegenden Einplatinencomputer arbeitende KI übernehmen. Durch eine Verarbeitung auf dem On-Board-Computer soll es möglich werden, die Fundstellen durch automatisiertes Anfliegen zu verifizieren. Die Verifikation soll es ermöglichen, von dem Objekt aus geringerer Entfernung Bildmaterial zu erhalten, welches dann auf dem Einplatinenrechner weitergehend verarbeitet wird, um genauere Aussagen über das Objekt zu erhalten. Dies könnte die Treffergenauigkeit gegenüber einer Post Processing Verarbeitung erheblich verbessern, da bei der Post Processing Methode nur auf Bildmaterial aus größerer Entfernung zurückgegriffen werden kann.

2 Grundlagen

2.1 Thermografie

Infrarot Thermographie ist im weiteren Sinne die Messung von elektromagnetischer Strahlung und deren Abbildung in fotoähnlicher Weise. Der technische Einsatzbereich ist sehr groß. Jener erstreckt sich von der Lebensmittelanalyse [Osb06] über Kartographie [Wal20] bis hin zur zerstörungsfreien Werkstoffanalyse [Qia20].

Elektromagnetische Strahlung kann als ein Fluss von Photonen verstanden werden. Photonen sind masselose Partikel, welche sich wellenförmig unter Lichtgeschwindigkeit ausbreiten. Im elektromagnetischen Spektrum befindet sich Infrarotstrahlung oberhalb des für das menschliche Auge sichtbaren Bereichs. Sie ist also langwelliger und erstreckt sich über einem Wellenbereich von etwa (0,7-1000 μm). [Ste05, Isr15]. Die Strahlungsintensität eines Objektes ist abhängig von der Oberflächentemperatur. Je höher die Temperatur, desto höher ist auch die abgegebene Strahlungsenergie [Usa14]. Die Infrarotstrahlung kann weiter unterkategorisiert werden, wobei nicht alle Bereiche in der Thermographie sinnvoll nutzbar sind. Es handelt sich hierbei nicht um feste Abgrenzungen, sondern eher Richtwerte:

- nahes Infrarot SWIR (0,8 - 3 μm)
- mittelwelliges Infrarot MWIR (3 - 5 μm)
- langwelliges Infrarot LWIR (7,5 - 14 μm)
- fernes Infrarot FIR (bis 1000 μm)

Aus diesen Bereichen hat mittelwelliges und langwelliges Infrarot für die Thermographie die größte Verwendung. Dies hat hauptsächlich zwei Gründe:

1. Die Strahlungsemission im nahen Infrarotbereich ist schlichtweg zu gering. Dies hat zur Folge, dass Sensoren wesentlich empfindlicher für gleiche Abbildungsqualität sein müssten.
2. Im Wellenlängenbereich von 5 bis 7 μm absorbiert die Atmosphäre fast alle und im Bereich von 0,4 bis 0,7 μm ca. 40 % der Strahlungsemissionen.

Deshalb arbeiten Infrarotmessgeräte meist im langwelligen oder mittelwelligen Infrarotbereich. Wobei mittelwellige sich eher für hohe und langwellige sich eher für Temperaturen um den Raumtemperaturbereich eignen [Usa14].

Für die Infrarotabbildung von Rehkitzen, deren Körpertemperatur üblicherweise um die 40°C (313 K) beträgt [Rog87], sollte der Infrarotsensor möglichst um den Wellenlängenbereich von 10 μm arbeiten. Wieso dies so ist wird klar, wenn man sich die planckschen Strahlungsspektren für unterschiedliche Temperaturen ansieht (Abb. 1). Für eine Temperatur von 300K wird das Maximum an Strahlung im langwelligen Infrarotbereich um 10 μm emittiert (Abb. 1 rote Kurve) [Sch18].

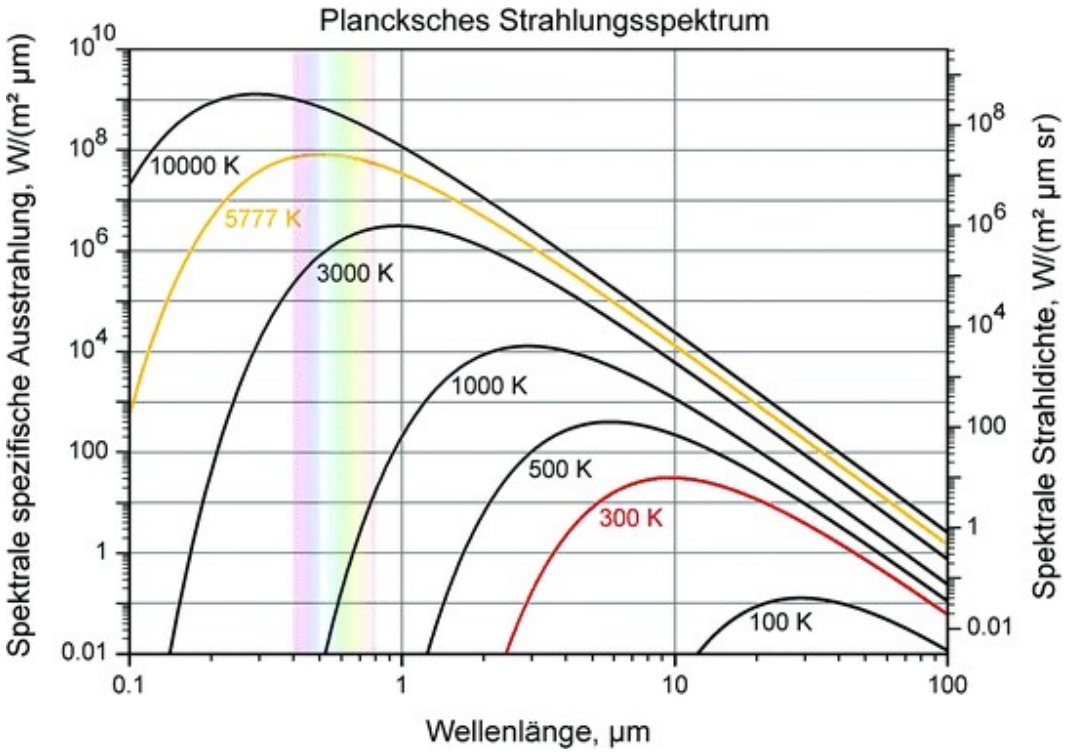

Abbildung 1: Plancksche Strahlungsspektren für unterschiedliche Temperaturen im doppelt logarithmischen Maßstab

Zur grafischen Abbildung wird die Strahlungsenergieverteilung über die Aufnahme einer Temperatur zugeordnet. Später kann über die Temperaturunterschiede innerhalb der Aufnahme ein Falschfarbenbild oder Graustufenbild erzeugt und somit für das menschliche Auge sichtbar gemacht werden. Im Gegensatz zur Nachtsicht ist eine externe Lichtquelle hierbei nicht nötig. Eine Abbildung von Tieren wird möglich, da 40-60 % des Wärmeverlustes bei Tieren im infraroten Strahlungsbereich geschehen [Ste05] und somit eine Abhebung von der meist kälteren Umgebung erfolgt.

Für den Anwendungsbereich der Rehkitzdetektion kommt eine Bildanalyse im sichtbaren Bereich nicht in Frage, denn die Rehkitze sind im hohen Gras oder Getreide sehr gut getarnt und selbst aus geringerer Entfernung, wie hier 10 m Flughöhe, kaum sichtbar (Abb. 2 links) [Isr15]. Im Spektralbereich heben sich die Kitze jedoch, besonders bei geringerer Sonneneinstrahlung, sehr gut von der Umgebung ab (Abb. 2 rechts). Nachts oder bei bedecktem Himmel ist eine Detektion also besonders einfach. Tagsüber, bei direkter Sonneneinstrahlung, ist die Erkennung eine größere Herausforderung, da sich dann auch kahle Stellen oder Maulwurfhügel in der Wiese erwärmen.

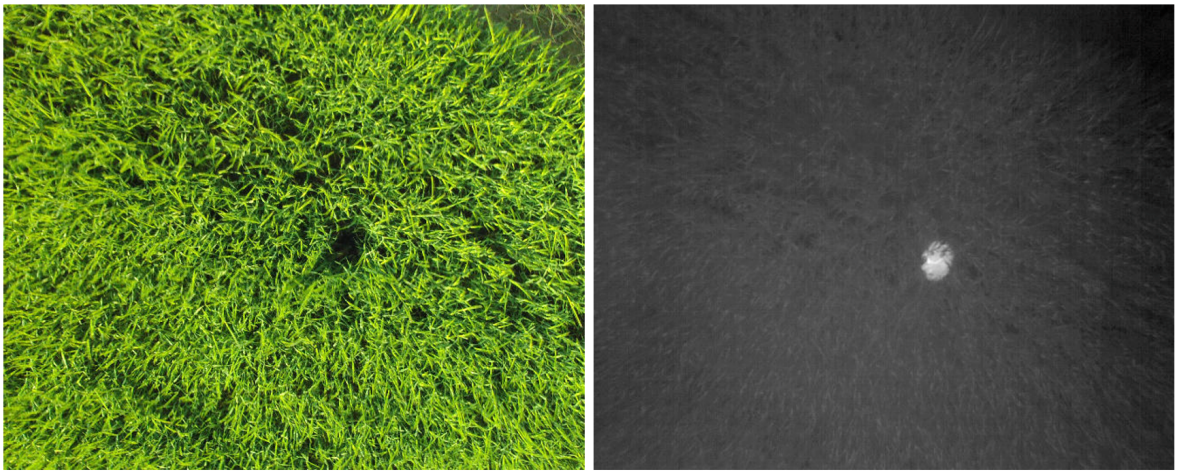

Abbildung 2: Vergleich eines Thermal (rechts) und Farbbildes (links) bei Tageslicht

2.2 Direkte Georeferenzierung

Bei der direkten Georeferenzierung handelt es sich um ein Verfahren, um mit einfachen Mitteln, also ohne zusätzliche Sensorik, aus Bilddaten Raumbezug zur Szene herzustellen. Hierbei werden Positions- und Orientierungsdaten der Kamera berücksichtigt. Da die Kamera jedoch über keine eigene Positionssensorik verfügt, muss die Sensorik des UAV-Gehäuses zur Hilfe genommen werden. Die Lage der Kamera relativ zum Trägergehäuse wird dann über die kardanische Aufhängung ermittelt [Isr15, Pfe12].

2.2.1 Orientierung und Position des Trägersystems

Für die Positions- und Lagebestimmung des Trägersystems wird, wie in Abb. 3 zu erkennen, die eingebaute Sensorik genutzt. Die Position des UAVs im Weltkoordinatensystem lässt sich über die Koordinaten (X, Y, Z) beschreiben [Pfe12]. Die räumliche Lage lässt sich über die drei Winkel (μ, γ, χ) darstellen. Hierbei ist μ der Rollwinkel, γ der Bahnwinkel gegenüber dem Horizont und χ der Azimut, also der Flugrichtungswinkel gegenüber Norden. Für die relative Höhe werden, um eine verbesserte Genauigkeit zu erreichen, Daten von GPS-Empfänger und Luftdrucksensor verwendet. Die Position in X und Y Richtung wird über das GPS-Modul ermittelt.

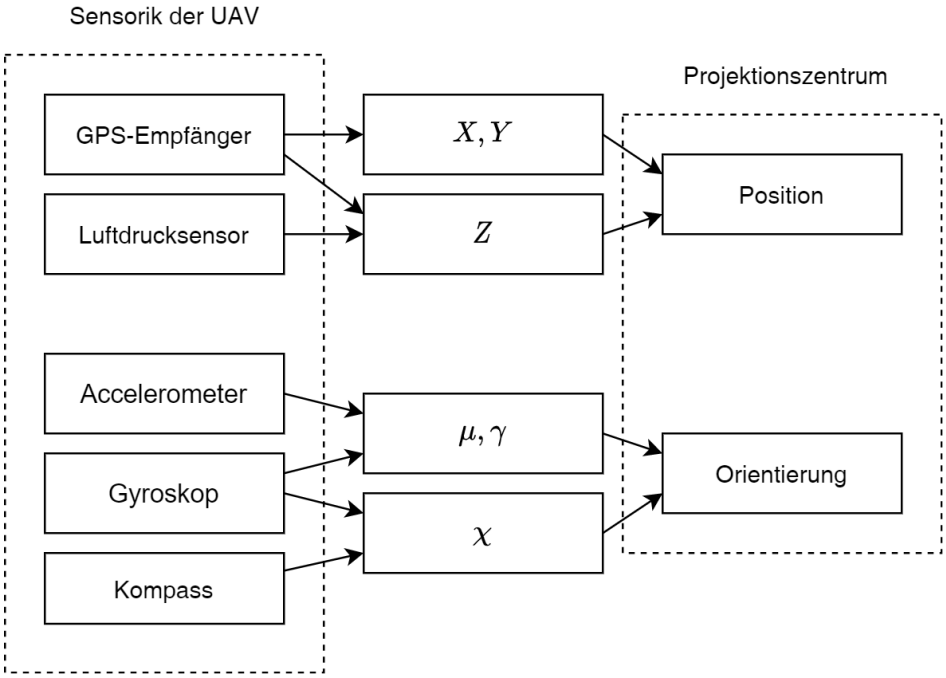

Abbildung 3: Sensorik des UAVs

Die Rotationswinkel können im Idealfall durch Integration der Roll- Gier- und Nickraten des Gyroskops ermittelt werden. Bei kleineren und preiswerten Sensoren würde aber durch die geringe Genauigkeit schnell ein großer Messfehler

bzw. ein Wegdriften vom eigentlichen Wert entstehen. [Pfe12, ES09] Deshalb werden zusätzlich Absolutwinkel für brauchbare Daten benötigt. Für Roll- und Nickwinkel wird dieser Absolutwinkel vom Accelerometer bestimmt und für den Gierwinkel vom Magnetometer (Kompass). Der Roll- und Nickwinkel wird über den Winkel zum senkrecht zur Erde zeigenden Gravitationsvektor (Nadir) gemessen (Abb. 4).

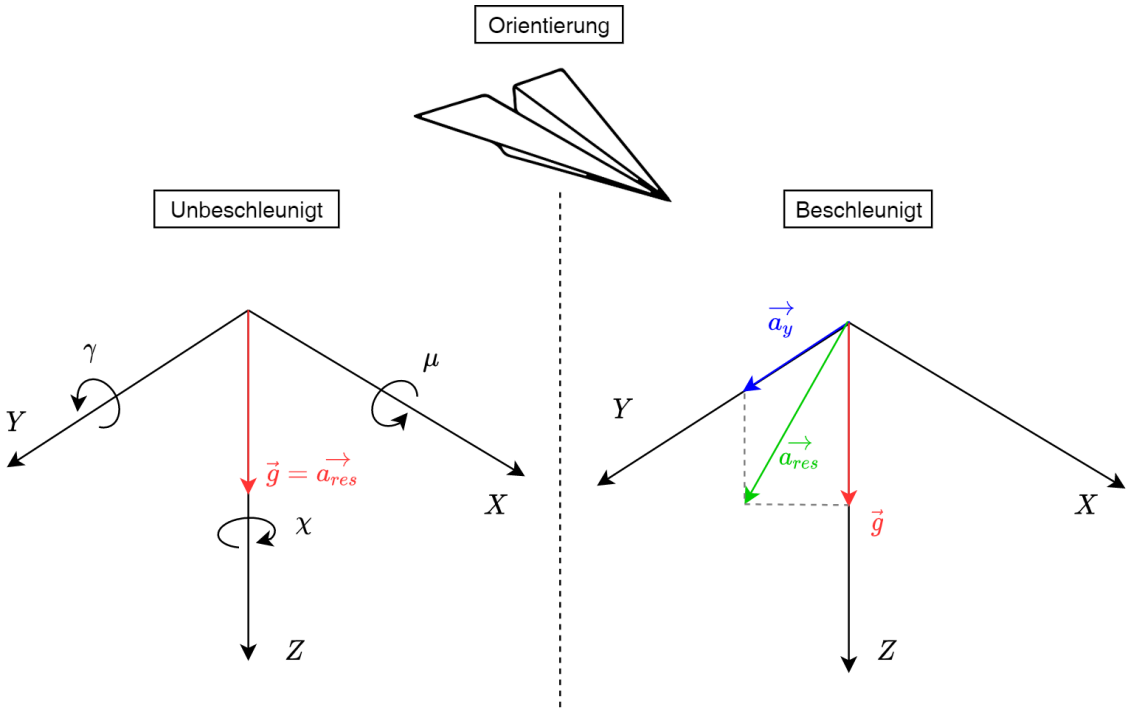

Abbildung 4: Orientierungswinkel und Koordinatensystem des UAVs bei unbeschleunigtem und beschleunigtem Flug

Bei nicht konstanter Fluggeschwindigkeit ist der Roll- und Nickwinkel jedoch fehlerbehaftet, da sich Beschleunigungen in X und Y Richtung mit dem Erdbeschleunigungsvektor vermischen und die resultierende Beschleunigung nicht mehr senkrecht nach unten zeigt. Aus diesem Grund werden die Orientierungswinkel μ, γ nur dann über den Gravitationsvektor bestimmt, wenn das UAV sich in einer unbeschleunigten Situation befindet. Während sich das UAV in einem Zustand befindet, bei der die Geschwindigkeit nicht kontant ist, werden die Orientierungswinkel über die Roll- und Nickraten des Gyroskops bestimmt [Tor18].

Bis jetzt wird davon ausgegangen, dass die Position und Orientierung von UAV-Gehäuse und Kamera deckungsgleich sind. Dem ist nicht so, da die Kamera schwenkbar montiert ist und konstruktionsbedingt nicht im Rotationsmittelpunkt des Trägergehäuses befestigt werden kann. Wie eine Transformation des Projektionszentrums auf die tatsächliche Position und Orientierung möglich ist, wird in 2.2.6 beschrieben.

2.2.2 Innere Orientierung

Die innere Orientierung beschreibt die innere Geometrie der Kamera und der Bildebene (Abb. 5). Die Bildebene ist das Koordinatensystem p, wobei x und y die Koordinaten des Hauptbildpunktes darstellen. Die Koordinaten des Zentralpunktes der Kamera sind c_x und c_y, diese müssen nicht zwangsweise im Zentrum der Bildebene liegen, sondern können durch Fertigungsgenauigkeiten der Kamera davon abweichen. Die Kammerkonstante c beschreibt den Abstand zwischen Projektionszentrum und Bildhauptpunkt, dieser wird häufig auch als Brennweite des Objektivs angegeben [Luh10].

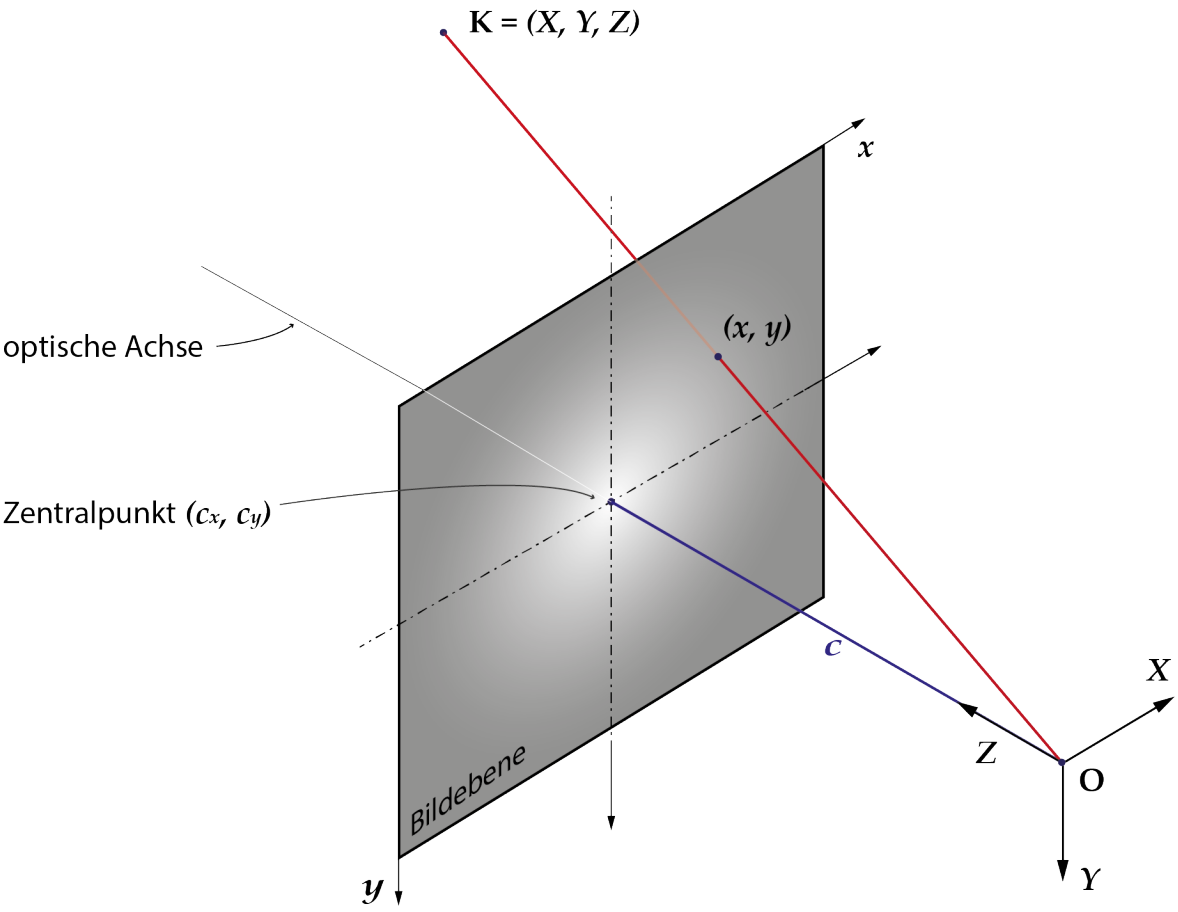

Abbildung 5: Innere Orientierung

In der Realität befindet sich das Projektionszentrum vor der Bildebene, sodass das Bild vertikal und horizontal gespiegelt abgelichtet wird. Für das verbesserte Verständnis wurde hier (Abb. 5) eine vereinfachte Darstellung mit dem Projektionszentrum hinter der Bildebene gewählt.

2.2.3 Verzeichnung des Bildes

Idealerweise handelt es sich bei der verwendeten Kamera um eine Lochkamera. Da eine Lochkamera keinerlei Linse benötigt, weist die Abbildung auch keine Verzeichnung auf (Abb. 6 [A]) [Sch09]. Moderne Kameras besitzen jedoch eine Blende, um bei unterschiedlichen Lichtstärken eine gleichbleibende Bildqualität zu gewährleisten. Eine Blende ist quasi auch nur ein Loch wie bei der Lochkamera, aber größenverstellbar. Durch die Verstellbarkeit der Öffnungsgröße lässt sich die Menge an einfallendem Licht regulieren. Um die Projektion bei größeren Blendenöffnungen scharf halten zu können, wird eine Linse benötigt, welche das Bild aber verzerrt (Abb. 6 [B, C]). Zusätzlich zur radialen Verzeichnung kommen noch andere Verzeichnungsformen vor, wobei die radiale am stärksten vertreten ist [Vil08].

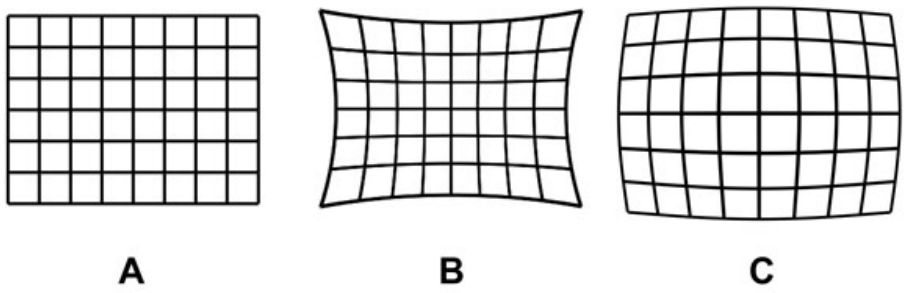

Abbildung 6: Beispiele von radialen Verzeichnungen - A: keine Verzeichnung, B: negative radiale Verzeichnung, C: positive radiale Verzeichnung

Beispielsweise kann durch die Fertigungstoleranzen in der Kamera- oder Objektivherstellung eine inexakte Ausrichtung der Linse entstehen. Durch diese Ausrichtungsfehler wird der Verzerrungsmittelpunkt auf der Bildebene zum Bildmittelpunkt verschoben.

2.2.4 Kompensation von Verzeichnungen

Da Verzeichnungen unvermeidbar sind, müssen diese softwaretechnisch herausgerechnet bzw. korrigiert werden. Meist wird hierzu das Brown-Conrady-Modell verwendet, welches radiale und tangentiale Verzeichnung korrigieren kann [Vil08, Dua71, Con19].

Für einen 3D-Punkt $\mathbf{K} = [X, Y, Z]^T$ gilt [Isr15]:

$$x' = \hat{x}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_2(r^2 + 2\hat{x}^2) + 2\,\hat{x}\,\hat{y}\,p_1 \tag{2.1}$$

$$y' = \hat{y}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1(r^2 + 2\hat{y}^2) + 2\,\hat{x}\,\hat{y}\,p_2 \tag{2.2}$$

$$x = c_x + x' f_x + y' s_{xy} \tag{2.3}$$

$$y = c_y + y' f_y \tag{2.4}$$

Mit:

$$\hat{x} = X/Z$$

$$\hat{y} = Y/Z$$

$$r = \sqrt{\hat{x}^2 + \hat{y}^2}$$

s_{xy}	schiefsymmetrische Koeffizient zwischen der x- und y-Achse
(c_x, c_y)	Koordinaten des Zentralpunktes der Kamera [Pixel]
(x, y)	projizierte Punktkoordinaten [Pixel]
(k_1, k_2, k_3)	Koeffizienten der radialen Verzeichnung
(p_1, p_2)	Koeffizienten der tangentialen Verzeichnung

2.2.5 Äußere Orientierung

Die äußere Orientierung (Abb. 7) beschreibt die Lage und Position des Projektionszentrums während der Aufnahme, bezogen auf den Aufnahmegegenstand. X_0, Y_0 und X_0 geben hierbei die Position des Projektionszentrums O im Umgebungskoordinatensystems m an. Die Lage, also die rotatorische Ausrichtung des Projektionszentrums werden durch die Winkel ω, ϕ und κ definiert. Während des Fluges wird die äußere Orientierung durch Nutzung von GPS und INS bestimmt. Bei, wie in diesem Fall verwendeten, Senkrechtaufnahmen ist der Abbildungsmaßstab als das Verhältnis von Kammerkonstante zu Flughöhe definiert [Luh10]:

$$\lambda = c/h \tag{2.5}$$

Die Flughöhe h entspricht in den meisten Fällen der Z-Koordinate (Z_0) im Objektraumkoordinatensystem.

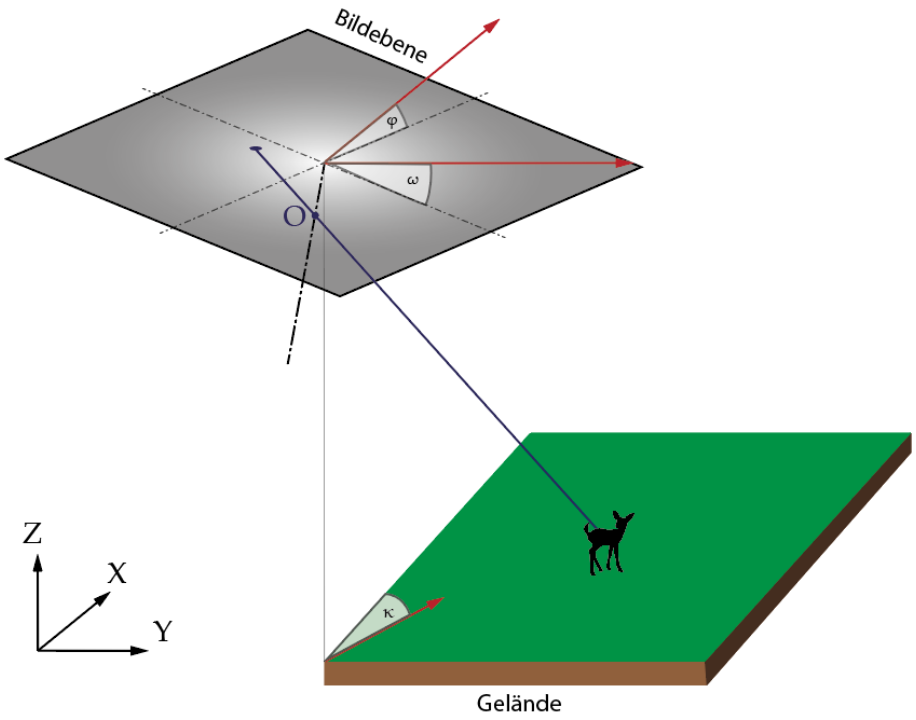

Abbildung 7: Äußere Orientierung

2.2.6 Transformation zwischen den Koordinatensystemen

Mit den Koordinatensystemen der Objektraumkoordinaten m und Bildkoordinaten p ergibt sich nach dem mathematischen Modell der Zentralprojektion die folgende Grundgleichung:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_m = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}_m + \lambda \mathbf{R}_p^m(\omega, \phi, \kappa) \begin{pmatrix} x - c_x \\ y - c_y \\ -c \end{pmatrix}_p \tag{2.6}$$

c	Kammerkonstante (Abb. 5)
(X_0, Y_0, Z_0)	Koordinaten des Projektionszentrums O bezogen auf das Objektraumkoordinatensystem m
$\mathbf{R}_p^m(\omega, \phi, \kappa)$	Rotation vom Bildkoordinatensystem p bezogen auf das Objektraumkoordinatensystem m

Hierbei wird die Einbauposition der Kamera jedoch noch nicht berücksichtigt.

Mit Berücksichtigung des Szenezustands und der Einbauposition- bzw. Winkel der Kamera wird nach Cramer [Cra06] die Georeferenzierung möglich. Diese lässt sich mit dem Koordinatensystem u der Flugplattform wie folgt beschreiben:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_m = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}_m + \mathbf{R}_m^u \left(\lambda \mathbf{R}_p^u \begin{pmatrix} x - c_x \\ y - c_y \\ -c \end{pmatrix} + \begin{pmatrix} \Delta X_C \\ \Delta Y_C \\ \Delta Z_C \end{pmatrix}_u - \begin{pmatrix} \Delta X_{GPS} \\ \Delta Y_{GPS} \\ \Delta Z_{GPS} \end{pmatrix}_u \right) \quad (2.7)$$

$\mathbf{R}_p^u$	Rotation vom Bildkoordinatensystem p in das Koordinatensystem u der Flugplattform
$\mathbf{R}_m^u$	Rotation von Koordinatensystem u in das Weltkoordinatensystem m
$(\Delta X_{GPS}, \Delta Y_{GPS}, \Delta Z_{GPS})$	Versatz der GPS-Antenne zum Ursprung des Koordinatensystems u des UAVs
$(\Delta X_C, \Delta Y_C, \Delta Z_C)$	Versatz des Projektionszentrums p zum Ursprung des Koordinatensystems u des UAVs

2.2.7 Reprojektion in GPS-Koordinaten

Zur Bestimmung der Koordinaten eines abgelichteten Bildpunktes wird nun das UTM-Koordinatensystem verwendet, da dieses im Gegensatz zu Längen- und Breitengraden orthogonal ist und somit eine Transformation erleichtert.

Bei der Transformation zwischen Kamera und UAV sowie zwischen UAV und Umgebungskoordinatensystem muss die Reihenfolge der Rotationsachsen beachtet werden. Für die Trägerplattform kann die Z, Y', X"- Konvention der Luftfahrt verwendet werden. Dies heißt, es wird erst um die Z- dann um die Y- und zuletzt um die X-Achse gedreht. Bei der Transformation der Kamera auf die Trägerplattform bestimmt die Reihenfolge der Drehungen die Aufhängung der Kamera. In diesem Fall wird eine X, Z', Y"-Konvention verwendet [Isr15].

Da die Kamera abstrakt auch als zentralperspektivische Abbildungsfunktion gesehen werden kann, ist eine Transformation in euklidischer Form nicht möglich. Für die Translation und Rotation muss aufgrund der fehlenden Parallelen in der projektiven Geometrie in homogenen Koordinaten gearbeitet werden. Hierfür werden die Rotationsmatritzen aus dem $\mathbb{R}^3$ um eine zusätzliche Dimension in

beiden Achsen erweitert und mit Nullen, bzw. auf der Diagonalen Einsen, aufgefüllt. Für eine Translation eines im $\mathbb{R}^3$ befindlichen Punktes gilt:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & \Delta X \\ 0 & 1 & 0 & \Delta Y \\ 0 & 0 & 1 & \Delta Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

Für die Transformationsmatrix $\mathbf{S}_c^m$ können nach [Isr15] alle Versatzstrecken der Kamera zum UAV bis auf den Versatz ΔX_C vernachlässigt werden. In Abhängigkeit der UAV-Position (X, Y, Z) und der Winkel (ω, ϕ, κ) lautet die Transformationsmatrix wie folgt:

$$\mathbf{S}_c^m = \mathbf{T}_m^u(X, Y, Z)\, \mathbf{R}_x(\omega)\, \mathbf{T}_u^p(\Delta X_C)\, \mathbf{R}_z(\kappa)\, \mathbf{R}_y(\phi)\, \mathbf{R}_p^u \tag{2.9}$$

Wobei die Terme Folgendes bedeuten:

$$\mathbf{T}_m^u(X, Y, Z) = \begin{pmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

$$\mathbf{R}_x(\omega) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\omega) & \sin(\omega) & 0 \\ 0 & sin(\omega) & cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.11}$$

$$\mathbf{T}_u^p(\Delta X_C) = \begin{pmatrix} 1 & 0 & 0 & \Delta X_C \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

$$\mathbf{R}_z(\kappa) = \begin{pmatrix} cos(\kappa) & -sin(\kappa) & 0 & 0 \\ sin(\kappa) & cos(\kappa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} cos(\phi) & 0 & -sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\phi) & 0 & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.14}$$

$$\mathbf{R}_p^u(\phi) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.15}$$

2.3 Haversine Formel

Im späteren Verlauf der Arbeit wird es nötig, den Abstand zwischen zwei GPS-Koordinaten bestimmen zu können. Zur Lokalisierung eines Punktes im GPS-Koordinatensystem werden häufig Längen- und Breitengrade verwendet. Über diese Kugelkoordinaten ist es möglich, die Lage eines Punktes auf der Erdoberfläche genau anzugeben. Die geografische Breite gibt den Winkel vom Äquator Richtung Norden (0° bis 90° Nord) bzw. Süden (0° bis 90° Süd) an. Längengrade laufen ausgehend vom Nullmeridian von 0° bis 180° Richtung Osten bzw. Westen.

Wenn die Erde nun als Kugel mit einem mittleren Radius von 6371 km angenommen wird, kann über die Haversine-Formel die kürzeste Distanz zwischen jeglichen Punkten auf der Erdoberfläche annähernd bestimmt werden. Mit den Längen- und Breitengraden zweier GPS-Koordinaten lautet die Haversine-Formel für die Distanz d:

$$d = 2\,R\,\arcsin\left(\sqrt{\sin^2\left(\frac{\Phi_2-\Phi_1}{2}\right)+\cos(\Phi_1)\,\cos(\Phi_2)\,\sin^2\left(\frac{\Psi_2-\Psi_1}{2}\right)}\right) \tag{2.16}$$

R Gemittelter Erdradius mit einem Wert von 6371 km

Ψ Längengrade der beiden Koordinaten zwischen welchen die Distanz bestimmt werden soll

Φ Breitengrade der beiden Koordinaten zwischen welchen die Distanz zu ermitteln ist

2.4 Künstliche Intelligenz und neuronale Netze

Bereits heute sind Computerprogramme bekannt, welche komplexe Aufgaben, für deren Lösung Intelligenz notwendig ist, besser und schneller bewältigen können als Menschen [Läm20]. Beispielsweise können bei Epileptikern bevorstehende Krampfanfälle anhand der Hirnaktivität vorhergesagt werden [Zha19]. Häufig handelt es sich um Probleme, bei welchen die Datengrundlage sich zu stark ändert oder die Eingabeparameter zu komplex sind. In diesen Fällen wäre eine klassische, wenn-dann Logikstrategie zu aufwändig oder gar unmöglich. In der klassischen Softwareentwicklung wird die Lösung zu einem Problem direkt als Lösungsalgorithmus einprogrammiert, wobei in der KI das Wissen getrennt der Verarbeitungskomponente steht [Läm20]. Häufig sind konventionelle, algorithmische Herangehensweisen aber schneller und zuverlässiger als der Einsatz von neuronalen Netzen. Der Begriff vom „neuronalen Netz “ entstammt der Biologie und beschreibt die Netzwerke der Nervenzellen in Gehirnen von Mensch und Tier. Gerade diesem System verdankt der Mensch das Vermögen verschiedenste intellektuelle und motorische Fähigkeiten zu erlernen und diese komplexen Umweltbedingungen anzupassen. Die Erkenntnis, dass diese Netzwerke an Nervenzellen für Dinge wie Bewusstsein, Gedanken und Wahrnehmung verantwortlich sind, wuchs erst um etwa 1900 [Ert16]. Die erste Basis für künstliche Intelligenz der neuronalen Netze wurde 1943 von McCulloch und Pitts gelegt, welche erstmals ein mathematisches Modell auf Grundlage des menschlichen Neurons erstellt haben [McC43].

2.4.1 Biologisches Vorbild

Um ein künstliches Modell von neuronalen Netzen verstehen zu können, ist eine grundlegende Kenntnis über menschliche Neuronen hilfreich. Ein menschliches Gehirn besteht aus etwa 100 Milliarden Neuronen, wobei alle Neuronen vereinfacht ähnlich aufgebaut sind (Abb. 8). Der Informationsfluss zum Neuron erfolgt über die Dendriten. Dendriten nehmen Impulse von vorherstehenden Neuronen auf und leiten diese an den Zellkörper weiter. Über Zellkern, Axon und Synapsen wird der Impuls dann an weitere Neuronen übertragen. Eine Reizweiterleitung findet nur statt, wenn die lokale elektrische Anregung stark genug ist. Eine neuronale Zelle kann hierbei mit bis zu zehntausend weiteren Neuronen vernetzt sein. Die Lernfähigkeit des Gehirns ergibt sich nun durch die veränderliche Leitfähigkeit der Synapsen, genauer der Neurotransmitter der Synapsen [Alb14]. Je mehr eine Synapse genutzt wird, umso mehr nimmt ihre Leitfähigkeit des Neurotransmitters zu. Umgekehrt nimmt die Leitfähigkeit ab, wenn die Synapse kaum aktiviert wird, was bis zum Absterben der Synapse führen kann. Somit wird die Lernfähigkeit von biologischen neuronalen Netzen klar. Wie aber durch diese

simplen Prinzipien intelligentes Verhalten möglich wird, ist bis heute fraglich [Ert16] [Meh97].

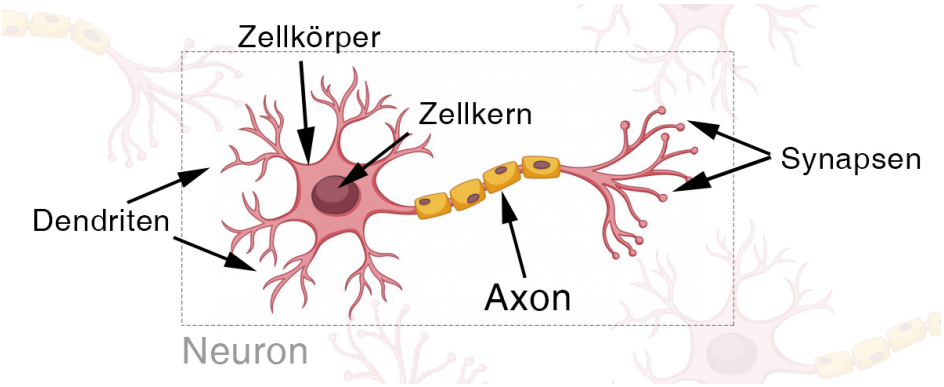

Abbildung 8: Vereinfachte Darstellung eines biologischen Neurons unter Verwendung von [Fre18]

2.4.2 Mathematisches Modell

Im Gegensatz zum biologischen Modell, arbeitet das simulierte nicht kontinuierlich, sondern mit einer vom Prozessortakt abhängigen Frequenz. Deshalb wird die stetige Zeitachse durch eine diskrete ersetzt. Das Neuron wird nun als verarbeitende Einheit gesehen, welche die eingehenden, gewichteten Werte zusammenfasst. Mittels der Aktivierungsfunktion wird unter Berücksichtigung eines Schwellenwertes eine Ausgabe des Neurons bestimmt. Um eine hohe Leistungsfähigkeit zu erreichen, wird wider Erwarten auf komplexe Algorithmen innerhalb der Neuronenfunktionen verzichtet. Die hohe Leistung wird erst durch ein Zusammenschalten vieler einfacher neuronalen Elemente bewerkstelligt.
Ein formales Neuron (Abb. 9) besteht aus folgenden Werten bzw. Funktionen:

- **Eingabeinformationen** $x_1 ... x_i ... x_n$: Sie sind die Eingabeinformationen, welche beispielsweise aus vorherstehenden Neuronen stammen können.
- **Propagierungsfunktion** net_j: Über die Propagierungsfunktion werden alle Eingabewerte mit unterschiedlicher Gewichtung w_{ij} aufsummiert.
- **Aktivierungsfunktion** f_{act}: Die Aktivierungsfunktion bestimmt unter der Berücksichtigung eines Schwellenwertes θ_j den neuen Zustand des Neurons.
- **Ausgabefunktion** f_{out}: Sie ermittelt aus zum aktuellen Zustand des Neurons den Ausgabewert x_j, welcher an nachstehende Neuronen weitergeleitet wird und dort wieder als Eingabeinformation auftritt.

Die Variablen des Zustandes (auch Aktivierung) a_j und des Schwellenwertes θ_j sind in einem lokalen Speicher enthalten.

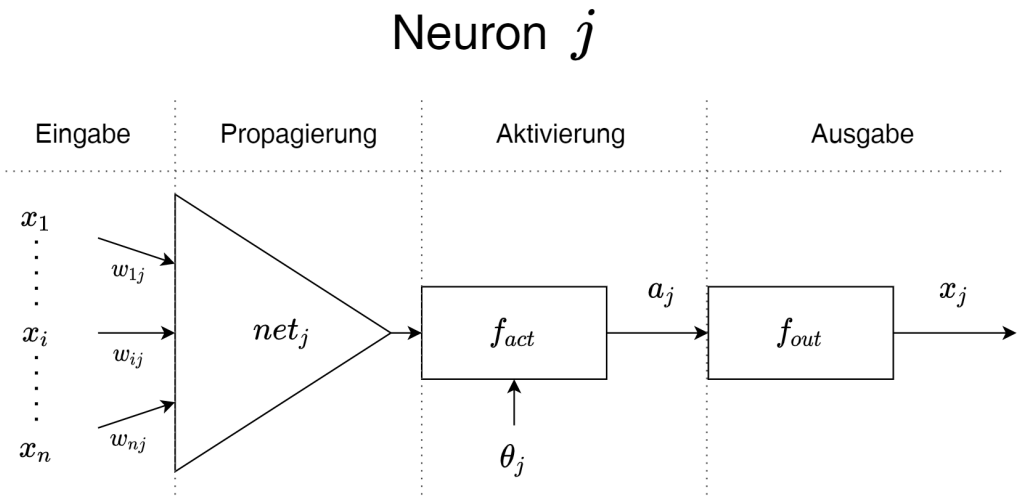

Abbildung 9: Mathematisches Modell eines Neurons

Eingabe und Propagierung Mithilfe der Propagierungsfunktion wird eine Vielzahl an Eingabewerten zu einem Wert zusammengefasst. Meist werden hierzu alle Eingabewerte gewichtet aufsummiert. Somit ist die Propagierungsfunktion der Form:

$$net_j = \sum_{i=1}^{n} w_{ij} \cdot x_i \tag{2.17}$$

net_j	Ausgabewert der Propagierungsfunktion
x_i	Eingabewert
w_{ij}	Gewichtung des Eingabewertes

Aktivierungsfunktionen Bei der Aktivierungsfunktion handelt es sich um eine Schaltfunktion, welche den Zustand des Neurons bestimmt. Sie hängt vom Ausgabewert der Propagierungsfunktion net_j und einem Schwellenwert θ_j ab. Die Aktivierungsfunktion entspricht also der Form:

$$f_{act} = f(net_j, \theta_j) \tag{2.18}$$

θ_j	Schwellenwert

Im einfachsten Fall ergibt die Aktivierungsfunktion die Identität der Eingabe ($f_{act} = net_j$). Dies kann aber zu Konvergenzproblemen in der Dynamik neuronaler Systeme führen, da die Identität nicht begrenzt ist. Deshalb wird häufig eine Schwellenwertfunktion zur Steuerung der Aktivierung eingesetzt. Sie ist die schärfste sigmoide Funktion. Für die Schwellenwertfunktion gilt:

$$f_{schwelle} = \begin{cases} 0, & falls\ net_j < \theta_j \\ 1, & sonst \end{cases} \tag{2.19}$$

Die Unstetigkeit der Schwellenwertsfunktion ist für einige Lernverfahren von Problem. Ein Glätten der Funktion ist aber möglich, indem man beispielsweise einen Tangens hyperbolicus oder eine Logistisch-Sigmoide-Funktion (Abb. 10 links) einsetzt [Sha17].

Für *Convolutional Neural Networks* wird meist eine Gleichrichterfunktion, auch als ReLU-Funktion bekannt, zur Aktivierung eingesetzt. Die Gleichrichterfunktion begrenzt lediglich den negativen Bereich. Sie liefert für den positiven Wertebereich die Identität und für den negativen Bereich 0 (Abb. 10 rechts).

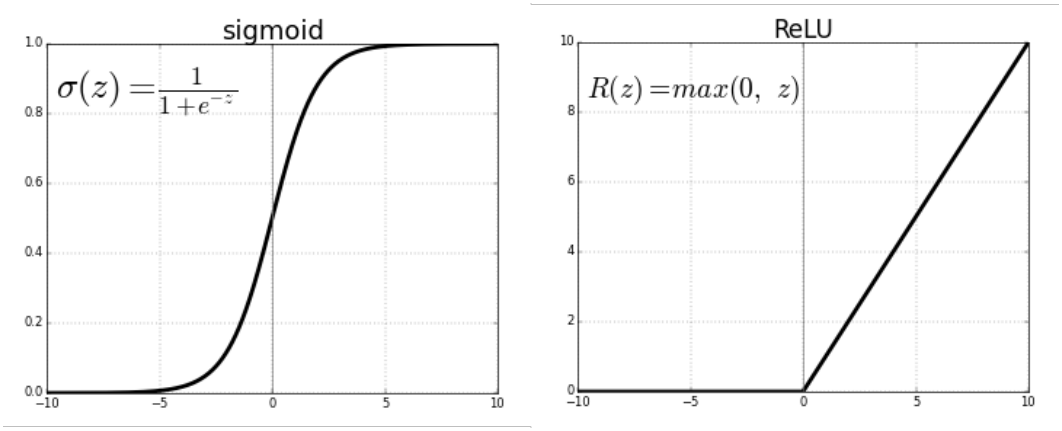

Abbildung 10: Sigmoide- und ReLU-Funktion im Wertebereich von -10 bis 10

Ausgabefunktion Über die Ausgabefunktion kann schließlich noch gesteuert werden, welchen Wert das Neuron in Abhängigkeit der Aktivierung ausgibt. In den meisten Fällen ist die Ausgabe gleich der Aktivierung. Also:

$$x_j = f_{out}(a_j) = a_j \tag{2.20}$$

a_j Aktivierungszustand des Neurons

2.5 Evaluation neuronaler Netze

Um zu bestimmen, wie gut eine KI performt ist es notwendig, eine passende Maßzahl für die Bewertung der KI zu erhalten. Für diese Metrik wird im Rahmen von CNNs oft von *Precision*, *Recall* und *Accuracy* gesprochen, teilweise auch von *Sensitivity* oder *Specificity*. [Kol18] Im weiteren Verlauf wird auf die, für diesen Fall zur Evaluation genutzten Werte, *Precision* und *Recall* eingegangen. Durch diese mathematisch definierten Maßzahlen sind Rückschlüsse auf die Qualität einer Objekterkennungssoftware möglich. Zur Berechnung werden die in der Konfusionsmatrix (Tabelle 4) definierten Größen verwendet.

in Wirklichkeit / Klassifizierung	**vorhanden**	**nicht vorhanden**
detektiert	TP	FP
nicht detektiert	FN	TN

Tabelle 1: Konfusionsmatrix

Die Parameter bedeuten dabei:

TP	*True Positive*: Es ist etwas vorhanden und wurde detektiert
TN	*True Negative*: Es ist nichts vorhanden und wurde nicht detektiert
FP	*False Positive*: Es wurde etwas detektiert, obwohl nichts vorhanden
FN	*False Negative*: Es wurde nichts detektiert, obwohl etwas vorhanden

Precision Bei Precision (dt. Präzision) geht es darum, wie genau die KI arbeitet. Es wird im Verhältnis betrachtet, wie viele Objekte erkannt wurden und wie viele davon wirklich die zu erkennenden Objekte sind. Wurde eine KI also auf ein gewisses Objekt trainiert, erkennt gehäuft aber auch andere Objekte, ist die Präzision eher niedrig.

$$Precision = \frac{TP}{TP + FP} \tag{2.21}$$

Recall Der Recall (dt. Abruf) gibt Aufschluss darüber, wie oft ein Objekt übersehen wird. Betrachtet wird hierbei das Verhältnis von übersehenen Objekten zu richtig erkannten. Dies ist besonders für den betrachteten Fall sehr bedeutsam, da möglichst keine Kitze übersehen werden sollten [Sor17].

$$Recall = \frac{TP}{TP + FN} \tag{2.22}$$

2.5.1 Verwendetes neuronales Netz

Im Rahmen der Objekterkennung wird für diese Arbeit ein Faltungsnetz verwendet. Faltungsnetze (*engl. convolutional neural network*) sind zum aktuellen Wissenstand die am besten geeigneten neuronalen Netze zur Objekterkennung in Bildern [Raz14].

Über die letzten Jahre haben sich Faltungsnetze in diversen Bereichen der optischen Objekterkennung bewährt. Beispielsweise in der biometrischen Merkmalerkennung, wie man sie aus dem Mobiltelefon zum Schutz vor Fremdzugriff kennt [Ngu18].

Für die genaue Netzarchitektur zur Rehkitzerkennung hat sich M. Sorg bereits im Rahmen seiner Bachelorarbeit Gedanken gemacht und einige Messungen durchgeführt [Sor17]. Durch den Vergleich unterschiedlicher Verschaltungen und deren Evaluation konnte die Performanz optimiert sowie eine geeignete Architektur entwickelt werden.

Insgesamt besteht das Netz aus zehn Schichten. Jede Faltung verschiebt den Filter um einen Pixel. Bei der Maximum-Vereinigung werden die Kernel um jeweils zwei Pixel verschoben.

1. **Eingabeschicht**: 64x64 Pixel große Graustufenbilder fungieren als Eingabe
2. **Faltungsschicht**: 16 Filter mit einer Filtergröße von 5x5
3. **Maximum-Vereinigung**: Kernelgröße von 2x2
4. **Faltungsschicht**: 32 Filter mit einer Filtergröße von 3x3
5. **Maximum-Vereinigung**: Kernelgröße 2x2
6. **Faltungsschicht**: 64 Filter mit einer Filtergröße von 3x3
7. **Maximum-Vereinigung**: Kernelgröße 2x2
8. **Vollvermaschte Schicht**: 8 Neuronen
9. **Vollvermaschte Schicht**: 8 Neuronen
10. **Ausgabeschicht**: 2 Neuronen

Als Aktivierungsfunktion wird in allen Faltungsschichten sowie in den vollvermaschten Schichten eine ReLU-Funktion (Abb. 10) genutzt. In der Ausgabeschicht

wird zur Aktivierung eine Softmax-Funktion verwendet.

Für das Training wurden etwa 100.000 Patches mit Tieren und 2,6 Millionen Patches ohne Tiere verwendet. Um zu evaluieren, wie gut die automatische Detektion mit dem verwendeten neuronalen Netz funktioniert, wurde von M. Sorg der Recall und die Precision (2.5) für unterschiedliche Wetterbedingungen gemessen [Sor17]. Hierfür wurden exemplarisch sechs Flüge aus dem Jahr 2015 ausgewählt. Die Ergebnisse sind in folgender Tabelle aufgelistet.

Flug	1	2	3	4	5	6
Schwierigkeit	einfach	einfach	mittel	mittel	schwer	schwer
Bewölkung	10/10	10/10	5/10	3/10	3/10	0/10
Datum	17.05.	17.05.	28.06.	31.05.	02.06.	28.05.
Uhrzeit	10:57	10:15	09:38	10:04	11:08	11:37
Precision	0,82	0,64	0,23	0,45	0,48	0,74
Recall	1,0	1,0	0,97	0,95	0,94	0,87

Tabelle 2: Performanz des verwendeten neuralen Netzes

Die Zeit und Datumsangaben beziehen sich hierbei auf den Aufnahmezeitpunkt des getesteten Datensatzes. Da zu den Datensätzen keine Umgebungstemperaturen gemessen wurden, wird auf Daten der Wetteraufzeichnung von *Wetterkontor* in München zurückgegriffen. Die Temperatur lag für alle Aufnahmetage bei einem Mittelwert von etwa 15°C.

2.6 MAVlink

Bei *MAVlink* handelt es sich um ein Nachrichtenprotokoll zur Kommunikation zwischen UAVs und Bodenstationen, sowie zwischen dem Bordcomputer und angebrachter Komponenten [Mei13]. Ursprünglich entwickelt wurde *MAVlink* von Lorenz Meier und die erste Version erschien 2010 auf *Github*. Es kann genutzt werden, um beispielsweise Informationen über den Zustand des UAVs zu bekommen, oder auch Missionspunkte für das UAV zu übermitteln. Implementiert wurde *MAVlink* in einigen Programmiersprachen wie *Java*, *C++*, und *Python*.

3 Messsystem

Bei dem Trägersystem handelt es sich, wie auf Abb. 11 (links) zu sehen um einen Hexacopter, welcher mit zusätzlichen Komponenten ausgestattet ist. An Bord befindet sich zur Aufnahme des überflogenen Bereiches eine Infrarotkamera mit kardanischer Aufhängung. Die Aufhängung ist auf Gummipuffern so gelagert, dass möglichst wenig Vibrationen auf die Kamera übertragen werden. Zur Steuerung des UAVs und zur Kommunikation mit der Bodenstation (Abb. 11 rechts) wird ein *Pixhawk* Flugregler verwendet. Die Verarbeitung der Bilder und die Abspeicherung der Objektpositionen von Kitzen übernimmt ein *RaspberryPi* der dritten Generation.

Abbildung 11: Trägersystem und Bodenstation

3.1 Kamera

Bei der Kamera handelt es sich um eine Langwellen-Infrarot (LWIR) Wärmebildkamera des Herstellers *FLIR* (Abb. 12) [FLI19]. Die Auflösung der Kamera beträgt 320 auf 256 Pixel und die feste Brennweite hat einen Wert von 6,3 mm. Der Sensor arbeitet im langwelligen Infrarot zwischen 7,5 μm und 13,5 μm Wellenlänge. Somit deckt die Kamera genau den in 2.1 beschriebenen benötigten Wellenlängenbereich ab.

Abbildung 12: FLIR, Boson 320, 34° (HFOV) 6,3 mm

3.2 Verbindung RaspberryPi zu Pixhawk

Damit das UAV auf Entscheidungen der Bildverarbeitung durch den *RaspberryPi* reagieren kann, muss eine stabile Kommunikationsverbindung zwischen dem *RaspberryPi* und dem *Pixhawk* ermöglicht werden. Hierfür wird eine serielle Kommunikation zwischen den Komponenten aufgebaut, indem die Anschlüsse, wie in Abb. 13 zu sehen, verkabelt werden [Alk17]. Als Kommunikationsstandard wird das in Kapitel 2.6 beschriebene Protokoll *MAVlink* verwendet.

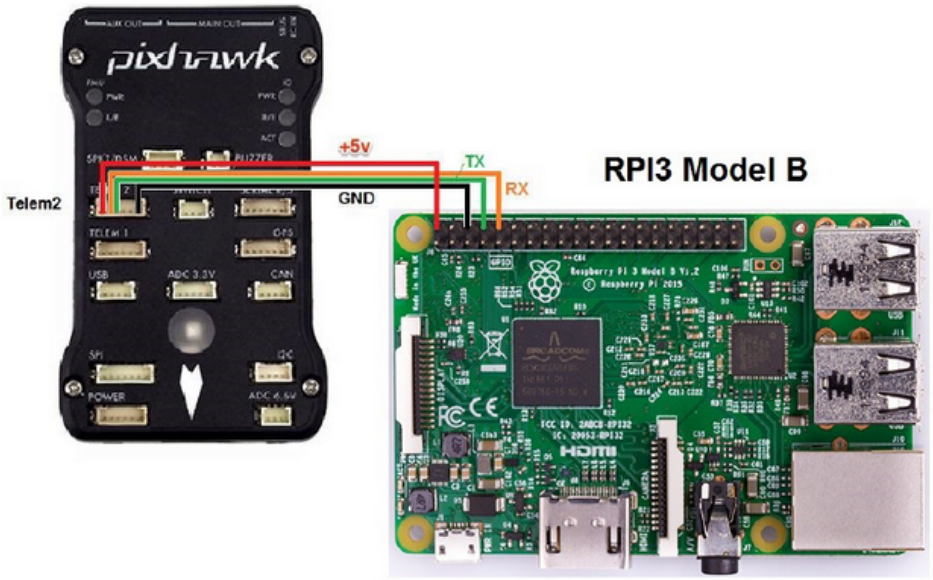

Abbildung 13: Verbindung zwischen Pixhawk und RaspberryPi

3.3 Künstliches Rehkitz

Da ein echtes Rehkitz aufgrund der schlechten Reproduzierbarkeit für Versuche mit dem System nicht in Frage kommt, wird eine künstliche Wärmequelle benötigt, welche einem realen Kitz in Größe und Temperatur möglichst ähnlich sein

sollte. Verwendet wurde deshalb ein 3D-gedrucktes Kunststoffgehäuse mit eingelassenem Heizwiderstand (Abb. 14). Mit den Abmaßen von 15 x 17 cm und einer Oberflächentemperatur von ca. 40 °C ist die Attrappe bei Infrarotaufnahmen einem echten Rehkitz sehr ähnlich. Für die Energieversorgung der Attrappe wird ein baugleicher Akku wie im Trägersystem genutzt.

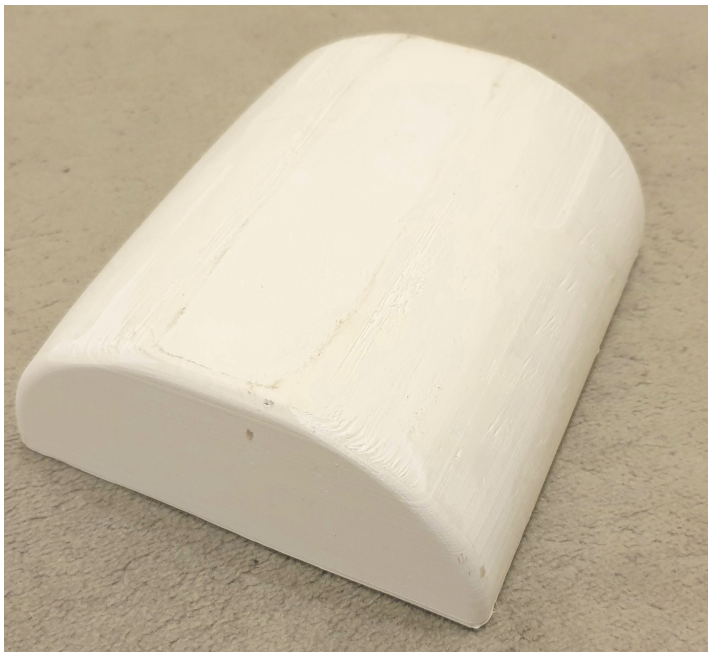

Abbildung 14: Foto vom künstlichen Kitz

4 Rehkitzerkennung und Lokalisierung

4.1 Missionsplanung

Zur Planung der Mission können unterschiedliche Missionsplaner zur Hilfe genommen werden. Eine mobile Version ist beispielsweise *QGroundControl*. Der Vorteil ist, dass die Missionsplanung direkt am Einsatzort und auf dem Smartphone vorgenommen werden kann.

Zum automatisierten Überflug über das abzusuchende Gebiet kann ein sogenanntes Survey angelegt werden. Hierbei müssen lediglich die Flughöhe über Grund, der Abstand zwischen den Flugpfaden und die Eckpunkte des Gebietes definiert werden. Für den Abstand zwischen den Flugpfaden ist die Schwadbreite S der Kamera ausschlaggebend ($S = 2\tan(FOV/2)h$). Die Kamera verfügt über einen HFOV von 34°. Bei einer Flughöhe h von 40 m beträgt die Schwadbreite also 24,5 m. Zur Sicherheit sollte beidseitig jedoch eine Überlappung von 10% eingeplant werden, wodurch der Abstand zwischen den Flugpfaden auf etwa 20 m reduziert wird. *QGroundControl* ermittelt mit den eingegebenen Parametern schließlich die optimale Flugroute (weiße Linien im grünen Bereich Abb. 15). Die Flugroute kann dann nach der Planung an den Flugregler des Trägersystems übertragen werden.

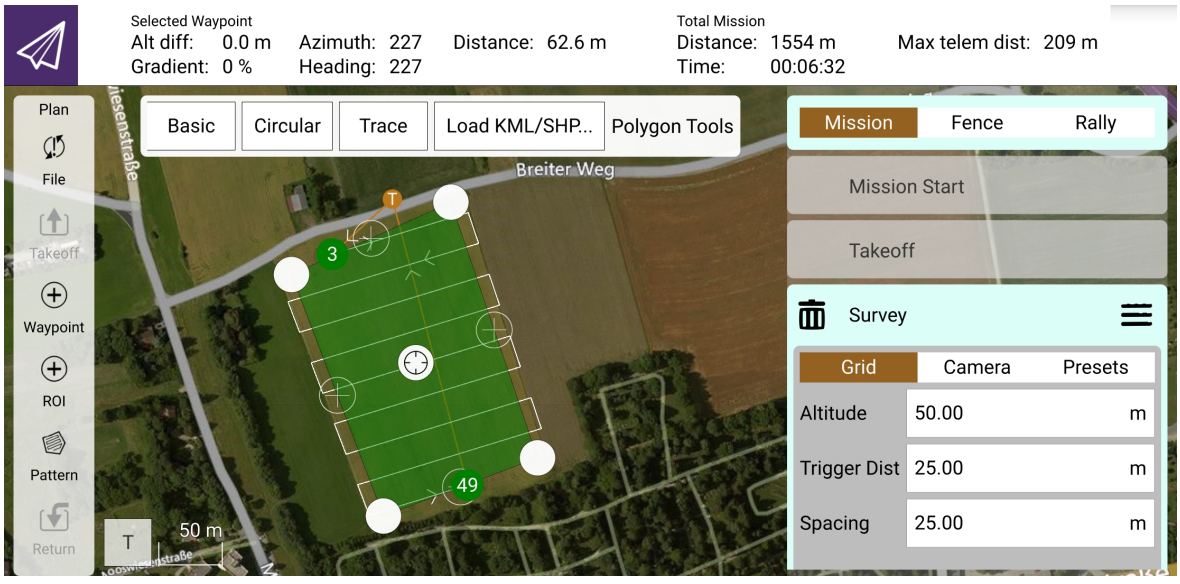

Abbildung 15: Bildschirmfoto einer Missionsplanung in QGroundControl

4.2 Missionsdurchführung

Am Einsatzort angekommen, wird das UAV in der Nähe des in 4.1 erstellten Bereiches positioniert. Nach manuellem Starten der Mission fliegt das UAV automatisiert die in der Missionsplanung (4.1) festgelegten Punkte ab und tätigt Infrarotbilder vom überflogenen Gebiet. Um direkt auf einen POI (mögliches Kitz) reagieren zu können, wird das Bild unmittelbar nach der Aufnahme auf dem *RasperryPi* analysiert.

4.3 Vorbereitung der Bilder für das neuronale Netz

Zunächst werden die Infrarotbilder für das neuronale Netz, wie in Abb. 16 zu sehen, vorbereitet. Die Größe des Bildes und die daraus resultierende Anzahl der Patches hängt von der Auflösung der Kamera ab. Hierbei gibt es im wesentlichen zwei unterschiedliche: 320x256 Pixel Sensor oder 640x512 Pixel Sensor. Für die Verarbeitung und Georeferenzierung spielt diese Auflösung keine Rolle, lediglich die Anzahl der Bildflicken unterscheidet sich zwischen 80 bzw. 20, wodurch die Performanz des Programmablaufes mit der Auflösung variiert.

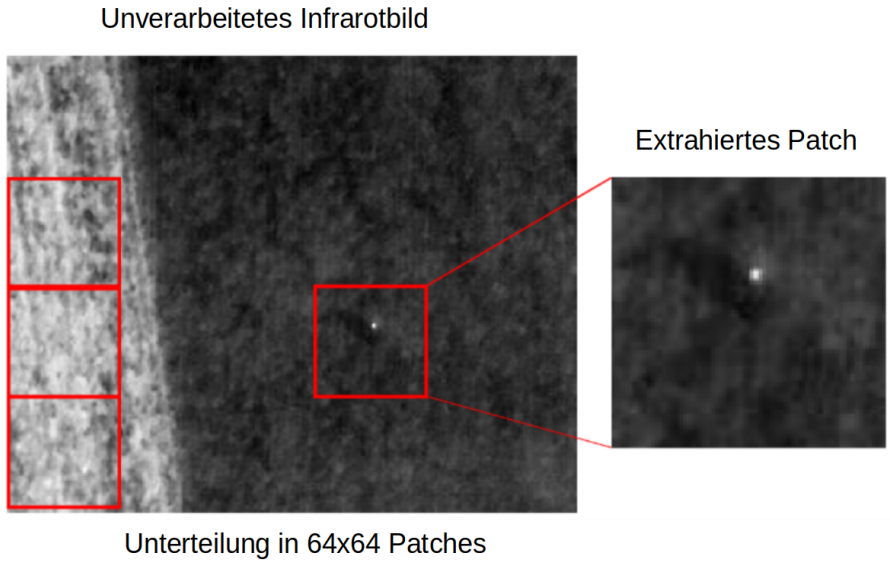

Abbildung 16: Patchextrahierung aus unverarbeitetem Infrarotbild

Da das Netz mit 64x64 Pixeln großen und normierten Bildflicken trainiert wurde [Sor17], sollte es auch mit einem Tensor der gewohnten Dimension verwendet werden [Wu20]. Aus diesem Grund wird das zu verarbeitende 320x256 Pixel große Bild in 20, 64x64 Pixel große Flicken zerlegt. Anschließend werden alle Pixel normiert, wodurch jedem Pixel ein Wert zwischen 0 und 1 zugeordnet wird. Hierbei entspricht 0 schwarz, 1 weiß und alle 253 Abstufungen dazwischen einer

Mischung, also einem Grauton. Der entstandene Tensor wird nun dem neuronalen Netz für eine Abschätzung übergeben.

Der Programmablauf der Bildvorbereitung ist in folgender Abbildung (Abb. 17) nach ISO-5807 [Int85] dargestellt.

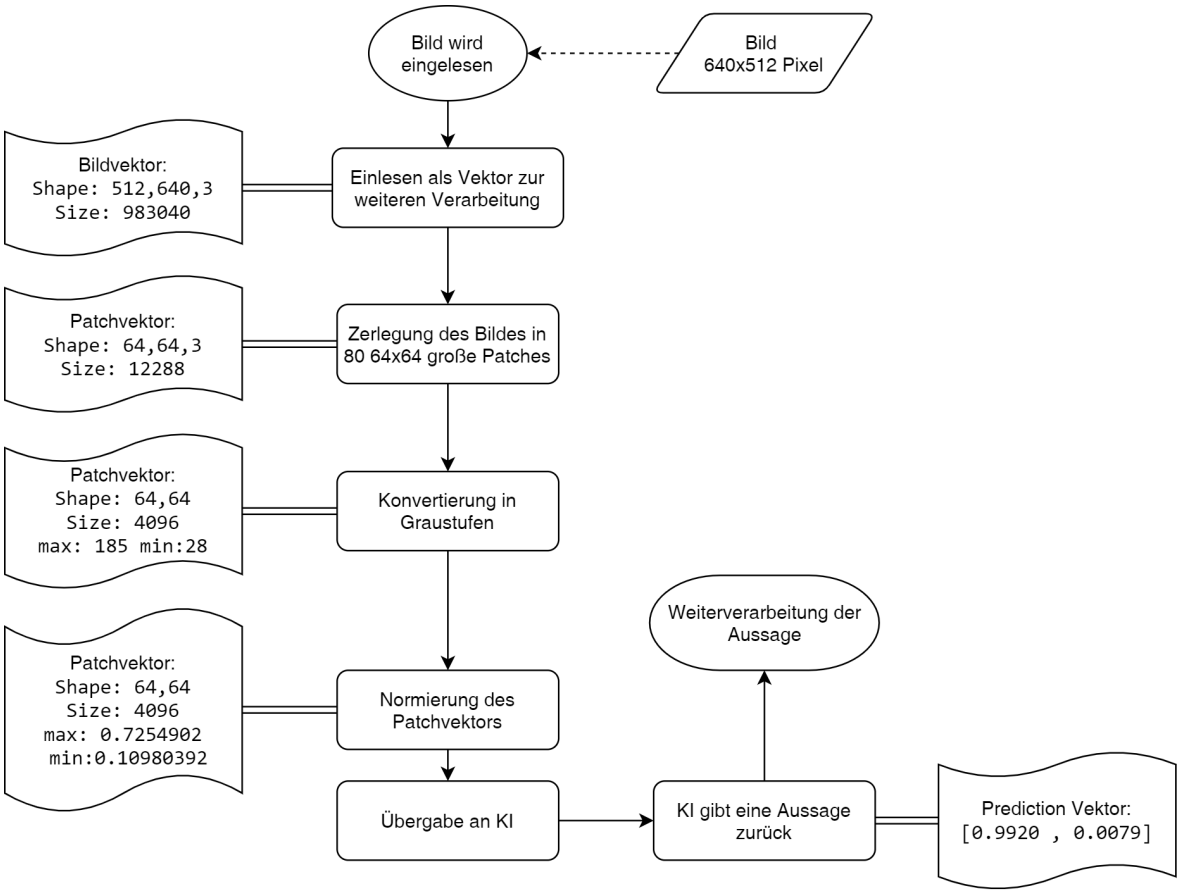

Abbildung 17: Vorbereitung und Übergabe der Bilder an das neuronale Netz

Beim Unterteilen des Bildes in Patches kann es jedoch in seltenen Fällen dazu kommen, dass der Bereich, auf dem das Kitz zu sehen ist, zerteilt wird. Somit könnte die Wärmesignatur auf mehrere Patches verteilt werden. Mit einer mittleren Kitzgröße von 4x4 Pixeln und der Patchgröße von 64x64 Pixeln würde die Wärmesignatur bei 744 Randpixeln zerteilt werden. Somit liegt die Wahrscheinlichkeit für eine Teilung bei 18 %. Kitze im Randbereich der Bildebene sind unproblematisch, da durch die Flugroute bereits eine 10-prozentige Überlappung der Ausleuchtungszone vorgegeben ist.

Die Wahrscheinlichkeit für eine Zerstückelung der Wärmesignatur sinkt jedoch immens, wenn man den Algorithmus in Zusammenhang mit der Fluggeschwindigkeit betrachtet. Bei der aktuell eingestellten Fluggeschwindigkeit von 5 m/s ist ein Objekt zwangsläufig auf mehr als einem Bild sichtbar. Die Breite der Aus-

leuchtungszone beträgt in einer Flughöhe von 40 m 24,5 m, jedoch ist der Bildsensor nicht quadratisch, wodurch bei einem Breiten- zu Höhenverhältnis von 5:4 (320:256) die Tiefe der Ausleuchtungszone 19,6 Meter beträgt. Bei einer Bildaufnahmefrequenz von 1 Hz und einer Fluggeschwindigkeit von 5 m/s wird jedes Objekt auf mindestens 3 Aufnahmen sichtbar. Die Wahrscheinlichkeit, dass eine Wärmesignatur in allen dieser 3 Aufnahmen an einem Patchrand geteilt wird, liegt somit bei $0,18^3$ also 0,6 %. Dies könnte im ungünstigsten Fall also dazu führen, dass jedes zweihundertste Kitz übersehen wird.

Es gibt jedoch eine Möglichkeit sicher zu gehen, dass keine Wärmesignatur zerteilt wird. Hierfür müsste man die Patches so überlappen lassen, dass ein Kitz im Randbereich eines Patches nicht zerteilt wird, sondern auf mehreren Bildflicken erscheint. Um die Patchdimension von 64x64 Pixel beizubehalten, muss vertikal sowie horizontal ein zusätzliches Patch erstellt werden. Wie in Abbildung 18 zu sehen, würden sich die Bildflicken in X-Richtung um 12 Pixel überschneiden und in Y-Richtung um 16 Pixel. Da in horizontaler Richtung ein Rest bleibt, müssten hier die äußersten 2 Pixel mit schwarz aufgefüllt werden. Durch die erhöhte Anzahl der Patches hätte die Methode jedoch einen Overhead in der Laufzeit. Mit 30 statt 20 zu verarbeitenden Bildflicken wäre mit einem Laufzeitzuwachs von 50 % zu rechnen.

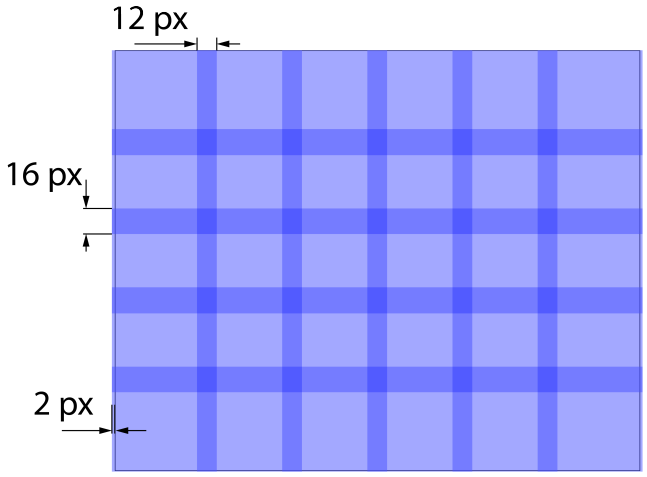

Abbildung 18: Möglichkeit zur Überlappung der Patches

4.4 Weiterverarbeitung mit der Aussage des neuronalen Netzes

Der Großteil des Bildes besteht aus uninteressanter Information, also in diesem Fall Feld oder Wiese [Gol19]. Um die interessanten Bildbereiche herauszufiltern und in diesen später die POIs zu bestimmen, wird das neuronale Netz zur Hilfe genommen. Nachdem die KI eine Entscheidung getroffen hat, wird ein Prediction Vektor erhalten. Dieser Vektor besteht aus 2 Einträgen $p = [x_1, x_2]$. Der erste Eintrag gibt nach Einschätzung der KI die Sicherheit an mit der sich kein Kitz auf dem Patch befindet. Analog steht der zweite Eintrag für die Sicherheit, dass sich auf dem Patch ein Kitz befindet. Die Summe der Vektoreinträge ergibt immer 1, also 100%. Somit handelt es sich laut Rückgabe um ein Kitz, wenn der Wert des zweiten Eintrages $p[1] = x_2 \geq 0.5$ entspricht.

Diese Aussage wird als Entscheidungsmerkmal verwendet ob, das erkannte Objekt angeflogen werden soll oder nicht. Zusätzlich zum Sicherheitswert wird aber noch die derzeitige Flughöhe und die Lage der Kamera beachtet. Dies ist notwendig, um unerwünschte Detektionen während Start- und Landevorgang zu verhindern. Eine Wärmesignatur wird deshalb nur dann weiterverwendet, wenn folgende Bedingungen erfüllt sind:

- Sicherheit der KI für das Objekt größer als 99 %
- Pitchwinkel der Kamera zwischen -100 und -80 °
- Rollwinkel der Kamera zwischen -10 und 10 °
- Flughöhe über 35 Meter

4.5 Bestimmung der Pixelposition eines erkannten Kitzes

Für eine spätere Koordinaten-Positions-Bestimmung ist es notwendig, die genaue Position in Pixelkoordinaten eines POIs zu kennen [Isr15]. Die in 4.4 vorgenommene Vorhersage sagt nur aus, ob sich auf dem Patch ein POI (Kitz) befindet oder eben nicht.

Innerhalb eines Patches ist das Kitz in den meisten Fällen das wärmste (hellste) Objekt. Wie in Abbildung 19 zu sehen, nimmt ein Kitz etwa eine Größe von etwa 4x4 Pixeln ein. Theoretisch müsste nun die Mitte der Pixelanhäufung bestimmt werden, jedoch entsprechen 4 Pixel über Grund einem Abstand von etwa 0,25 Metern. Somit liegt diese Abweichung immer weit unter der vom GPS-Empfänger erreichbaren Genauigkeit. Da jedem Pixel ein Wert zwischen 0 und 255 zugeordnet ist, kann die Position des Pixels mit dem maximalen Wert bestimmt werden.

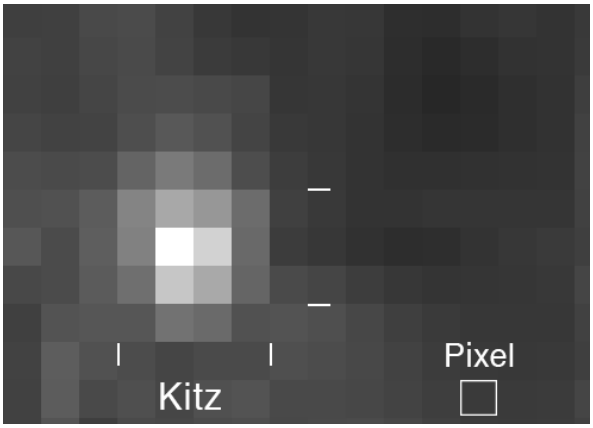

Abbildung 19: Bildausschnit mit Kitz aus einer Aufnahmehöhe von 40 Metern

Theoretisch wäre es möglich, dass neben dem Kitz noch ein weiteres helleres Objekt innerhalb eines Patches zu sehen ist. Diese Fälle werden aktuell nicht gesondert behandelt. Hierdurch könnte es somit zu fehlerhaft bestimmten Pixelpositionen kommen. Es gibt jedoch Möglichkeiten, diese Fehler zu filtern. Beispielsweise könnte ein sogenannter „Blob Detection“ Algorithmus implementiert werden, mit welchem sich die Fläche des Objektes berechnen ließe. Mit bekannter Flughöhe und mittlerer Kitzgröße wären somit größenmäßig unpassende Flächen herausfilterbar [Isr15].

Nachdem nun die relative Pixelposition (x^*, y^*) innerhalb eines Patches bekannt ist (Abb. 20), muss noch die globale Pixelposition (x, y) für das Bild bestimmt werden. Mit der Patchnummer i den Bildabmessungen (B, H) sowie den Patchabmessungen (P_B, P_H) sind diese nach folgender Formel ermittelbar:

$$x = s\, P_B + x^* \tag{4.1}$$

$$y = z\, P_H + y^* \tag{4.2}$$

Mit:

$$z = \left\lfloor \frac{i\, P_B}{B} \right\rfloor$$

$$s = \left\lfloor i - z\, \frac{B}{P_B} \right\rfloor$$

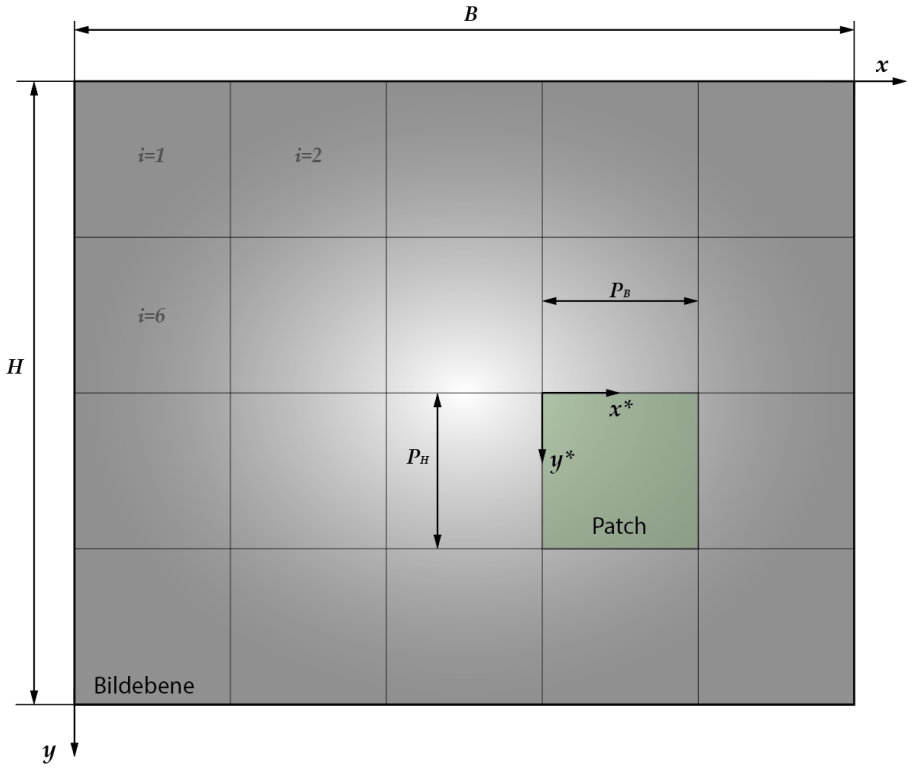

Abbildung 20: Bildebene und Bildflicken

4.6 Ermittlung der Fundstellenkoordinaten

Nach der Bestimmung der Pixelposition eines erkannten Kitzes (4.5) sind die Pixelkoordinaten bekannt. Diese können über die in 2.2 beschriebene Georeferenzierung genutzt werden, um die Position in Längen- und Breitengrad zu be-

stimmen. Zusätzlich zu den Pixelkoordinaten werden jedoch Informationen über den Szenenzustand sowie über Kalibrierungsparameter benötigt. Die Kalibrierungsparameter sind in einer Datei festgelegte Werte, welche Auskunft über Kameradaten wie die Brennweite der Kameralinse, aber auch Einbaufehlerwinkel zwischen Kameraaufhängung und UAV geben. Zum Szenezustand gehören die Zustandsdaten sowie die Position der Kamera.

Diese Szeneinformationen werden bei der Aufnahme eines Fotos in die Metadaten des Bildes abgespeichert. Zu diesen Informationen gehören beispielsweise GPS-Koordinaten vom UAV, Ausrichtung der Kamera, Flughöhe über Grund sowie der Aufnahmezeitpunkt. Da die Kamera jedoch linsenbedingt eine radiale Bildverzeichnung aufweist, muss diese korrigiert werden (2.2.4), da es sonst besonders im äußeren Bildbereich zu einem Fehler der Positionsbestimmung kommen würde.

Nachdem die Verzeichnung aus dem Bild korrigiert wurde, kann mittels der Transformationsmatrix $\mathbf{S}_c^m$ (Gleichung 2.9) die Position in GPS-Koordinaten bestimmt werden. Zur Schnittpunktbestimmung mit dem Geländemodell wird ein Raymarchingalgorithmus verwendet.

Für Überprüfungszwecke kann die Fundstelle mit den ermittelten Informationen als Bild lokal auf dem *RaspberryPi* abgespeichert werden. Die Fundstelle wird in der Aufnahme mit einem roten Kreis markiert. Neben den Fundstellenkoordinaten sind zusätzlich noch Informationen über die Pixelposition, sowie über Bildhelligkeit und Sicherheit der KI ablesbar. Eine beispielhafte Ausgabe ist in folgendem Bild (Abb. 21) dargestellt.

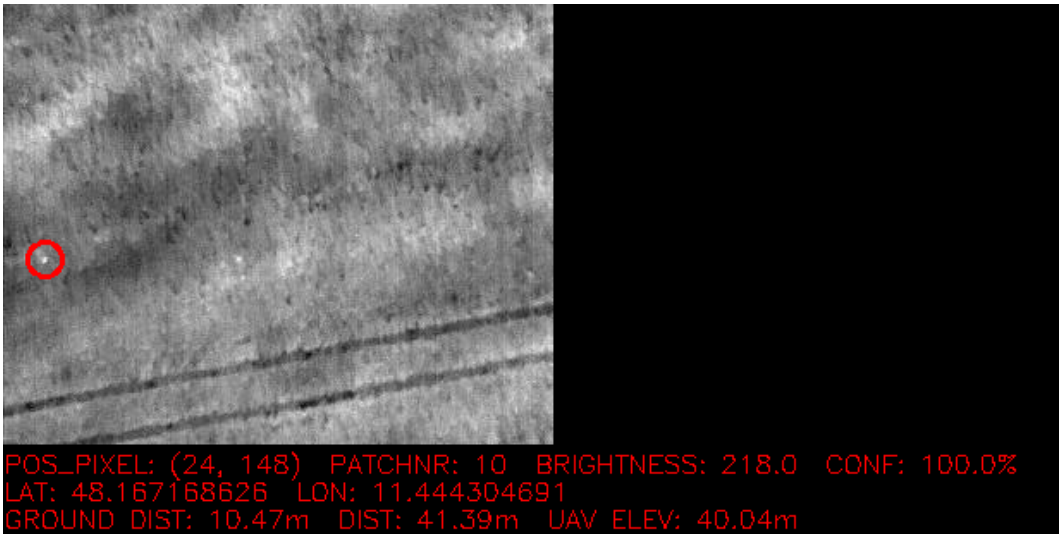

Abbildung 21: Grafische Ausgabe einer Fundstelle

4.7 Vereinigung naheliegender Fundstellen

Durch Ungenauigkeiten in der Sensorik des Trägersystems kommt es bei der Bestimmung der Fundstellenkoordinaten zu Abweichungen. Da die Kamera etwa sekündlich neue Bilder aufnimmt, wird ein Objekt durch die Überlappung auf mehreren Bildern erscheinen. Auch durch den vorgegebenen Flugpfad, welcher das ganze Feld abdecken soll, sind Überschneidungen nicht auszuschließen. Später sollen Punkte verifiziert (4.9), also im weiteren Sinne angeflogen werden. Es soll jedoch verhindert werden, dass identische Objekte mehrfach angeflogen werden und somit unnötig Zeit und Akkukapazität verschwendet wird.

In der Photogrammetrie ist dies ein bekanntes Problem und wird häufig durch sogenanntes „Image stitching“ gelöst [Sze06]. Beim Stitching werden die Bilder anhand von Bildmerkmalen, wie Konturen oder Kanten, so übereinandergelegt, dass ein großes Gesamtbild (Abb. 22) entsteht. In diesem einen Gesamtbild sind dann alle POIs enthalten und somit kann in diesem Bild pixelgenau gearbeitet werden.

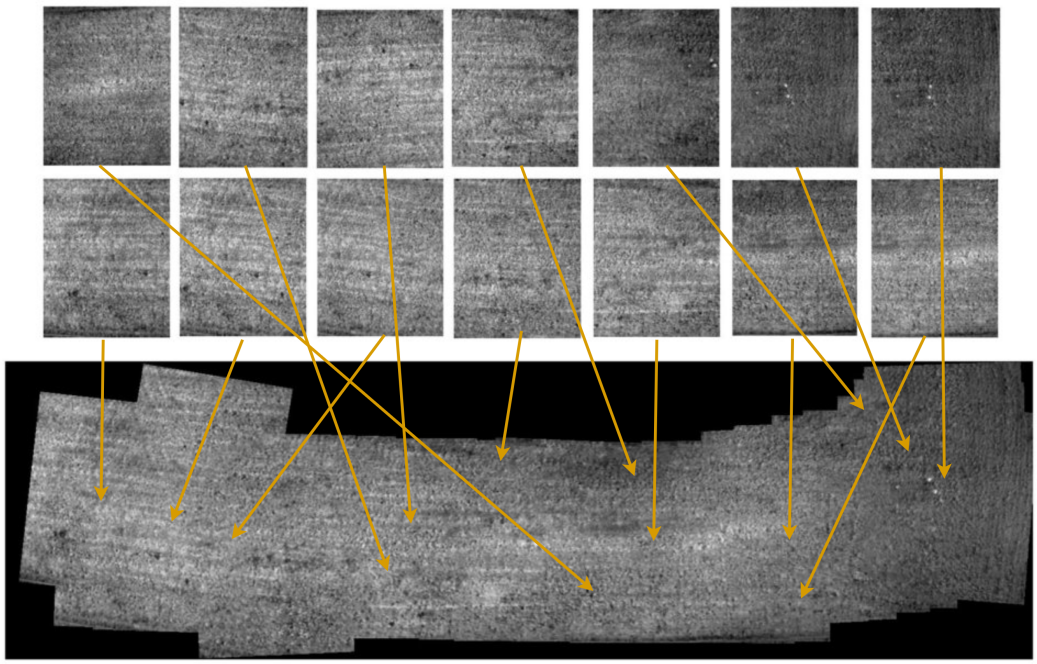

Abbildung 22: Grafische Darstellung eines Stitchingalgorithmus angewendet auf Bilder des fliegenden Wildretters

In dieser Arbeit wurde jedoch von einem On-Board-Stitchingalgorithmus Abstand genommen, dies hat hauptsächlich zwei Gründe:

1. **Performanz**: Da der *RaspberryPi* über begrenzte Rechenleistung verfügt und die implementierte Bildverarbeitung und KI die Prozessorleistung bereits an die Grenzen bringen, wäre mit einem zusätzlichen Stitchingalgorithmus die geforderte Bildaufnahmefrequenz von 1 Hz nicht erreichbar. Besonders wäre dieser Performanzeinbruch zu vernehmen, wenn die Bildgröße des Gesamtbildes zunimmt, was aber zwangsläufig passiert, wenn das UAV größere Bereiche abfliegt.

2. **Diversität der Aufnahmen**: Bei den Aufnahmen handelt es sich in den meisten Fällen um Wiesen von oben, somit haben alle Aufnahmen recht ähnliche Struktur. Zu sehen ist diese geringe Diversität in den Beispielbildern für den in Abb. 22 gezeigten Algorithmus. Auch Merkmale wie Fahrgassen, welche auf beispielsweise Weizenackern zu sehen wären, fehlen auf Nutzwiesen. Somit ist es für den Stitchingalgorithmus sehr schwer bis unmöglich, Konturen und Merkmale auf Bildern aus unterschiedlichen Aufnahmepositionen wiederzufinden.

Aus diesen genannten Gründen wurde eine Clusteranalyse implementiert, welche einen wesentlich geringeren Einfluss auf die Performanz hat. Bei dem entwickelten Clusteralgorithmus wird für jeden neuen POI geprüft, ob der Abstand zu einem Cluster innerhalb einer gewissen Toleranz liegt. Wenn dies der Fall ist wird der POI dem Cluster zugeordnet. Dieser Abstand wird mithilfe der in 2.3 beschriebenen Haversine-Formel berechnet. Sollte kein passendes Cluster gefunden werden, wird ein neues Cluster erstellt und initialisiert. Der genaue Ablauf des Algorithmus ist in folgendem Flussdiagramm schematisch dargestellt.

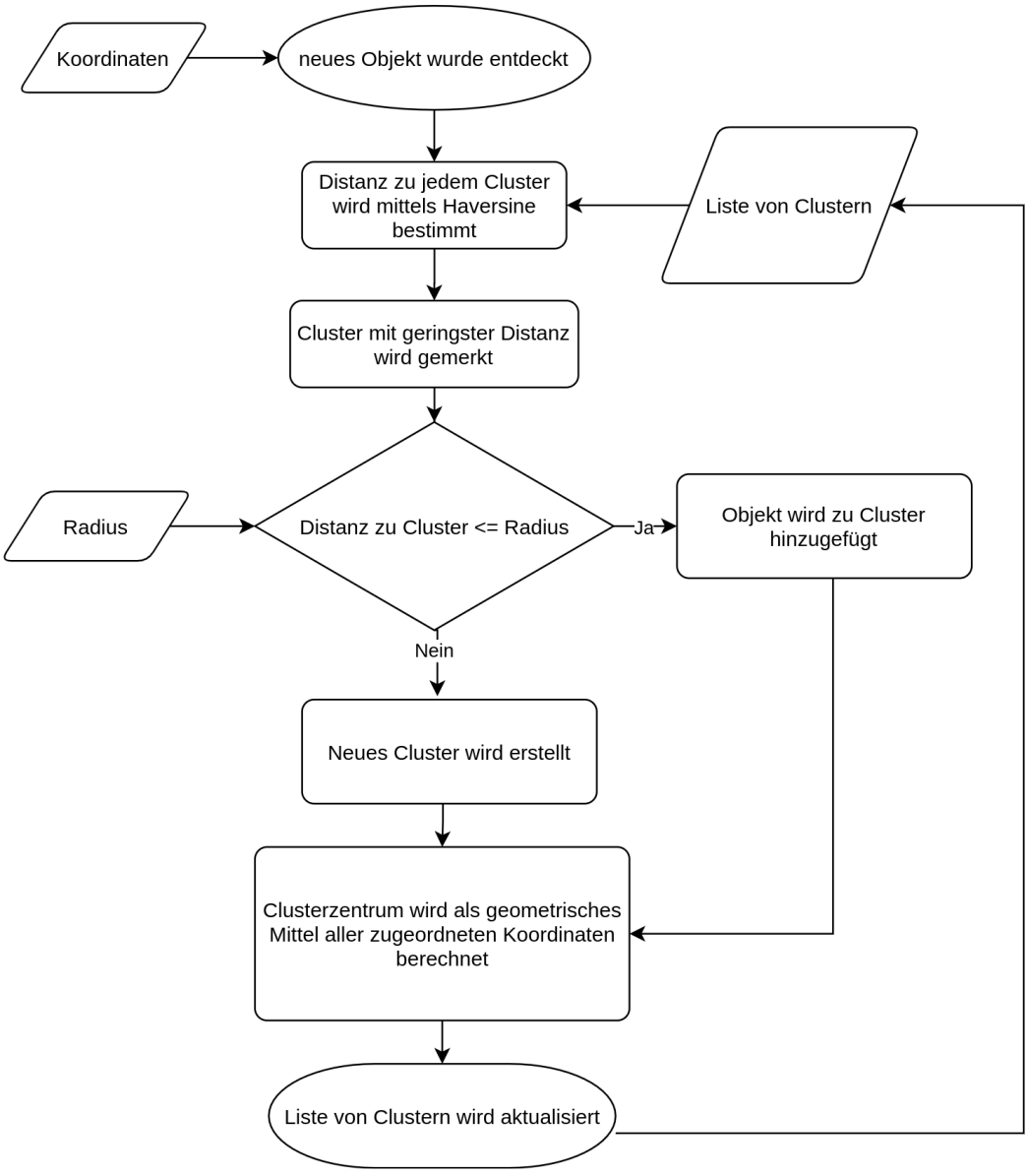

Abbildung 23: Ablauf der Clusteranalyse

Um zu überprüfen, ob der Clusteringalgorithmus wie gewünscht funktioniert, wurde ein Test in *PyGame* geschrieben. Die grafische Ausgabe des Tests ist in Abb. 24 zu sehen. Die farbigen Punkte entsprechen hierbei POIs, wobei Punkte der gleichen Farbe einem Cluster zugeordnet sind. Die schwarzen Quadrate stellen jeweils die Position des berechneten Zentrums der Anhäufungen dar. Der Radius des Kreises um den Mittelpunkt entspricht dem Radius des Clusters. Bei einer Überschneidung wird der neue Punkt zu dem naheliegenderen Verbund hinzugefügt. Wird ein Punkt außerhalb dieses Radius hinzugefügt, wird ein neues Cluster erzeugt.

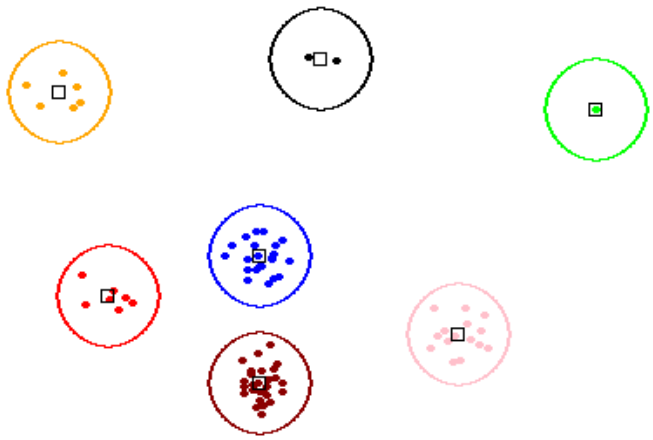

Abbildung 24: Grafische Ausgabe des in *PyGame* simulierten Tests für die Clusteranalyse

4.8 Anfliegen eines POIs

Im normalen Flugablauf befindet sich das UAV auf einer Flughöhe von 40 Metern. Für die Verifikation (4.9) ist eine genauere Begutachtung aus geringerer Entfernung nötig. Um dies zu ermöglichen, müssen dem *Pixhawk* Bordcomputer die neuen Koordinaten und die neue Flughöhe übermittelt werden. Das vom *RaspberryPI* ausgehende Kommando wird über serielle Schnittstelle mittels *MAVlink*-Protokoll (2.6) gesendet.

Normalerweise gibt es von *MAVlink* den Befehl `MAV_CMD_OVERRIDE_GOTO`, bei welchem die Mission unterbrochen wird und der als Parameter eingetragene Wegpunkt angeflogen wird. Während dieser Arbeit gab es mit dem Befehl jedoch Probleme, da *Ardupilot* den Befehl teilweise ignoriert hat. Außerdem muss sichergestellt werden, dass das UAV nach dem Anfliegen des Punktes senkrecht

wieder auf die Arbeitshöhe steigt, damit kein Bereich im abzusuchenden Gebiet auf zu niedriger Höhe überflogen wird. Mittels `MAV_CMD_OVERRIDE_GOTO` lässt sich dies nicht definieren, weshalb die gewünschte Funktion manuell implementiert wurde.

Bei der implementierten Methode wird, falls neue Wegpunkte hinzugefügt werden sollen, die gesamte auf dem *Pixhawk* befindliche Mission mit allen Missionsitems heruntergeladen und der Index des aktuell ausgeführten Wegpunktes wird ermittelt. Das Herunterladen erfolgt indem zunächst an den *Pixhawk* ein `MISSION_REQUEST_LIST` Befehl gesendet wird. Nach dem Empfangen des Befehls wird die Anzahl der auf dem *Pixhawk* gespeicherten Missionsitems übermittelt. Vom *RaspberryPI* können die Missionsitems dann mittels `MISSION_REQUEST_INT` für die Übertragung angefragt werden. Diese können schließlich einzeln vom *RaspberryPI* empfangen und gespeichert werden. Der genaue Ablauf des Missionsdownloads ist in Abb. 25 zu sehen [Wil18].

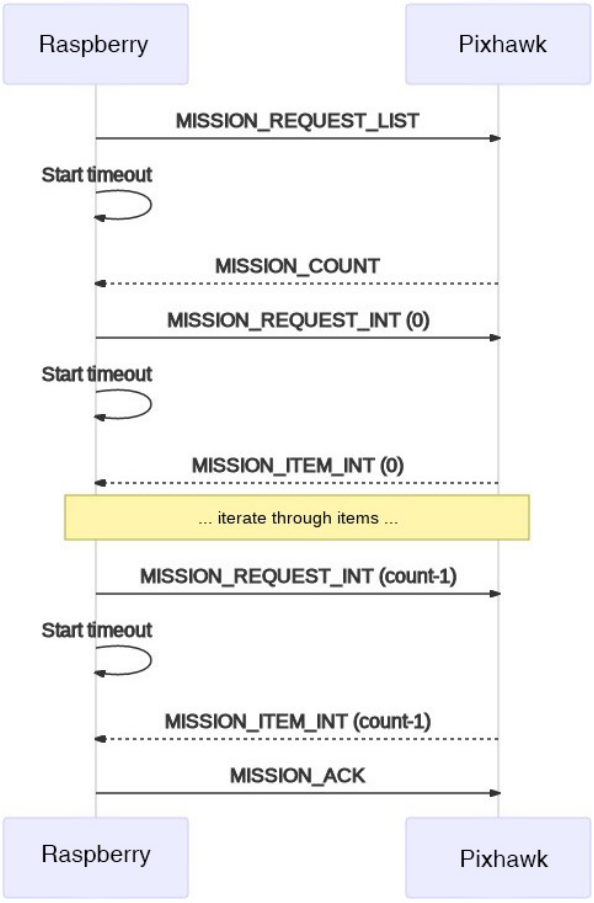

Abbildung 25: Herunterladen einer Mission

Der aktuelle Missionsstand, also bei welchem Index sich das UAV derzeit befindet, wird über die Nachricht `MISSION_CURRENT` abgefragt.

Nachdem die Liste der Missionen erhalten wurde, können am gewünschten Index die neuen Punkte eingefügt werden. Ein Missionsitem verfügt dabei über folgende Parameter:

`target_system`	ID des Systems = 0
`target_component`	ID der Komponente = 0
`seq`	Index zu welchem der Wegpunkt ausgeführt wird
`frame`	Koordinatensystem des Wegpunktes = 3
`command`	Kommando das am Wegpunkt ausgeführt wird = 16
`current`	{0, 1} ob der Wegpunkt der aktuelle ist
`autocontinue`	{0, 1} automatisch zum nächsten Wegpunkt = 1
`param1`	`hold`: Haltezeit am Punkt = 0
`param2`	`accept_radius`: Akzeptierter Radius = 0
`param3`	`pass_radius`: Abstand zum Wegpunkt = 0
`param4`	`yaw`: Gierwinkel am Wegpunkt = NaN
`x, y`	Längengrad, Breitengrad des Wegpunktes
`z`	relative Höhe über Grund

Das Hochladen der Mission funktioniert ähnlich wie das Herunterladen. Lediglich muss der *Raspberry* dem *Pixhawk* anfangs mitteilen, wieviele Missionsitems zu empfangen sind. Dies geschieht über die Nachricht `MISSION_COUNT`.

Nach dem Hochladen muss dem *Pixhawk* noch mitgeteilt werden, dass der gerade eingefügte Wegpunkt sofort ausgeführt werden soll. Dies erfolgt über den Befehl `MISSION_SET_CURRENT`.

Im Laufe der Arbeit wurden noch weitere Methoden zur *MAVlink* Kommunikation implementiert, wobei nicht alle Anwendung gefunden haben. Ein Überblick der in *Python* verwirklichten Methoden zusammen mit einer kurzen Beschreibung sind in folgender Tabelle aufgelistet.

Methode	**Beschreibung**
`arm`	Bereitet das UAV auf einen Start vor und aktiviert die Motoren
`disarm`	Deaktiviert die Motoren
`takeoff`	Lässt das UAV auf eine vorgegebene Höhe steigen
`set_mode`	Der Modus des Bordcomputers wird auf den angegebenen Modus gesetzt zB. *guided*
`log_all`	Alle empfangenen Nachrichten werden in ein Logfile geschrieben
`mission_start`	Startet wenn verfügbar die Mission
`mission_download`	Lädt die auf dem Bordcomputers befindliche Mission herunter
`mission_upload`	Lädt eine Mission auf den Bordcomputer hoch
`insert_mission_wp`	In einer lokalen Mission kann ein Wegpunkt an einem definierten Punkt eingefügt werden
`mission_set_current`	Aktives Missionsitem wird unterbrochen und die Mission wird bei einem definierten Missionsitem fortgesetzt
`gcs_heartbeat`	Sendet einen *heartbeat* an den Bordcomputer
`get_info`	Fragt grundlegende Informationen vom Bordcomputer ab
`connection_alive`	Prüft, ob der Letzte *heartbeat* vom Bordcomputer innerhalb einer gewissen Zeit empfangen wurde
`get_sat_count`	Aktuell sichtbare Satelliten und *GPS-fix-type* wird abgefragt
`get_location`	Aktualisiert die ermittelte Position vom UAV
`statustext`	Schreibt den letzten *statustext* in die Konsole
`waiter`	Unterbricht den Softwarefluss für eine bestimmte Zeit, sendet aber derweil einen *heartbeat*

Tabelle 3: Implementierte Methoden zur vereinfachten Kommunikation über *MAVlink*

Um die mittels *MAVlink* integrierten Methoden zu testen ohne dabei das Trägersystem zu gefährden, wurde eine sogenannte „Software in the Loop“-Umgebung verwendet. Hierbei wird das System mittels eines Modells, welches sich wie der *Pixhawk* verhält auf einem beliebigen Computer simuliert. Verwendet wurde hierfür der *Ardupilot* Simulator für *DroneKit*. Um das Verhalten des UAVs auf die Kommandos zu visualisieren, wurde *MissionPlanner* verwendet. Damit *MissionPlanner* die Daten des Simulators auslesen kann, müssen die ausgegebenen Nachrichten zunächst mittels *MAVProxy* weitergeleitet werden. Im *MissionPlanner* lässt sich dann auf einer Satellitenkarte das Verhalten vom UAV beobachten. Daten über Lage, Geschwindigkeit oder Flughöhe sind ebenso ablesbar.

4.9 Verifikation einer möglichen Kitzfundstelle

Als Verifikation wird während dieser Arbeit das genauere Betrachten einer Fundstelle bzw. einer Ansammlung von Fundstellen (Cluster) bezeichnet. Hierbei werden Aufnahmen von der Fundstelle aus geringerer Entfernung verwendet. Aus dieser Distanz nimmt ein mögliches Kitz wesentlich mehr Pixel der Aufnahmeabbildung ein. Bei einer Flughöhe von 40 Metern wird ein durchschnittliches Kitz mit 4 Pixeln (2x2 px) abgebildet (Abb. 26 links). Bei einer Höhe von beispielsweise 5 Metern beträgt die durchschnittliche Abbildungsgröße 256 Pixel (16x16 px) (Abb. 26 rechts). Der Informationsgewinn wird durch die geringere Entfernung proportional gesteigert. Für eine Distanzverminderung um Faktor 8 steigt die vom Objekt erhaltene Information also auch um Faktor 8.

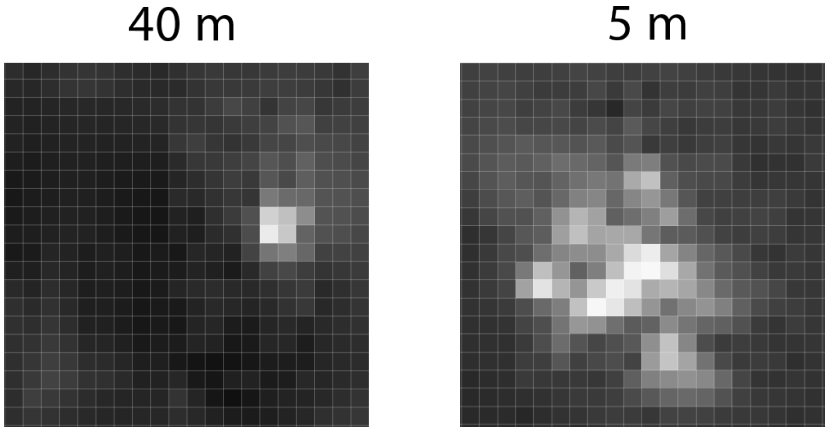

Abbildung 26: Informationsgewinn bei einer Flughöhe von 5 m gegenüber 40 m

Um eine Verifikation eines Clusters zu ermöglichen, verfügt jedes Cluster-Objekt über zwei Verifikationsvariablen, `verified` und `is_kitz`. Beim Erstellen eines Clusters werden beide Variablen mit dem booleschen Wert `false` initialisiert. Nach jeder Bildaufnahme werden alle Cluster überprüft ob die `verified` variable auf `true` steht. Ist dem nicht so, wird das Zentrum des Clusters bei geringerer Höhe angeflogen und es werden Bilder auf dieser geringeren Höhe aufgenommen. Diese „genaueren“ Bilder werden dann der KI übergeben, um zu prüfen, ob sich in dem Aufnahmebereich ein Kitz befinden sollte. Sollte ein Kitz in dem Bereich erkannt werden, wird im angeflogenen Cluster die `is_kitz`-Variable auf `true` gesetzt. Wenn kein Kitz in dem Bereich erkannt wurde, wird die `is_kitz`-Variable entsprechend auf `false` gesetzt.

Nach jeder Verifikation wird auf den lokalen Speicher die Liste von Clustern als *JSON* abgespeichert. Eine beispielhafte Ausgabe ist folgend gezeigt:


```
    "POI_Clusters": [
        {
            "center": [
                48.059289827,
                11.224187621
            ],
            "is_kitz": true,
            "points": [
                [
                    48.059289827,
                    11.224187621
                ]
            ],
            "verified": true
        },
        {
            "center": [
                48.059173429,
                11.22364389
            ],
            "is_kitz": false,
            "points": [
                [
                    48.059180911,
                    11.22362909
                ],
                [
                    48.059165947,
                    11.22365869
                ]
            ],
            "verified": true
        }]
```

Der gesamte Verifikationsablauf, ausgehend vom entdeckten Kitz, ist in folgendem Flussdiagramm nochmals genauer dargestellt:

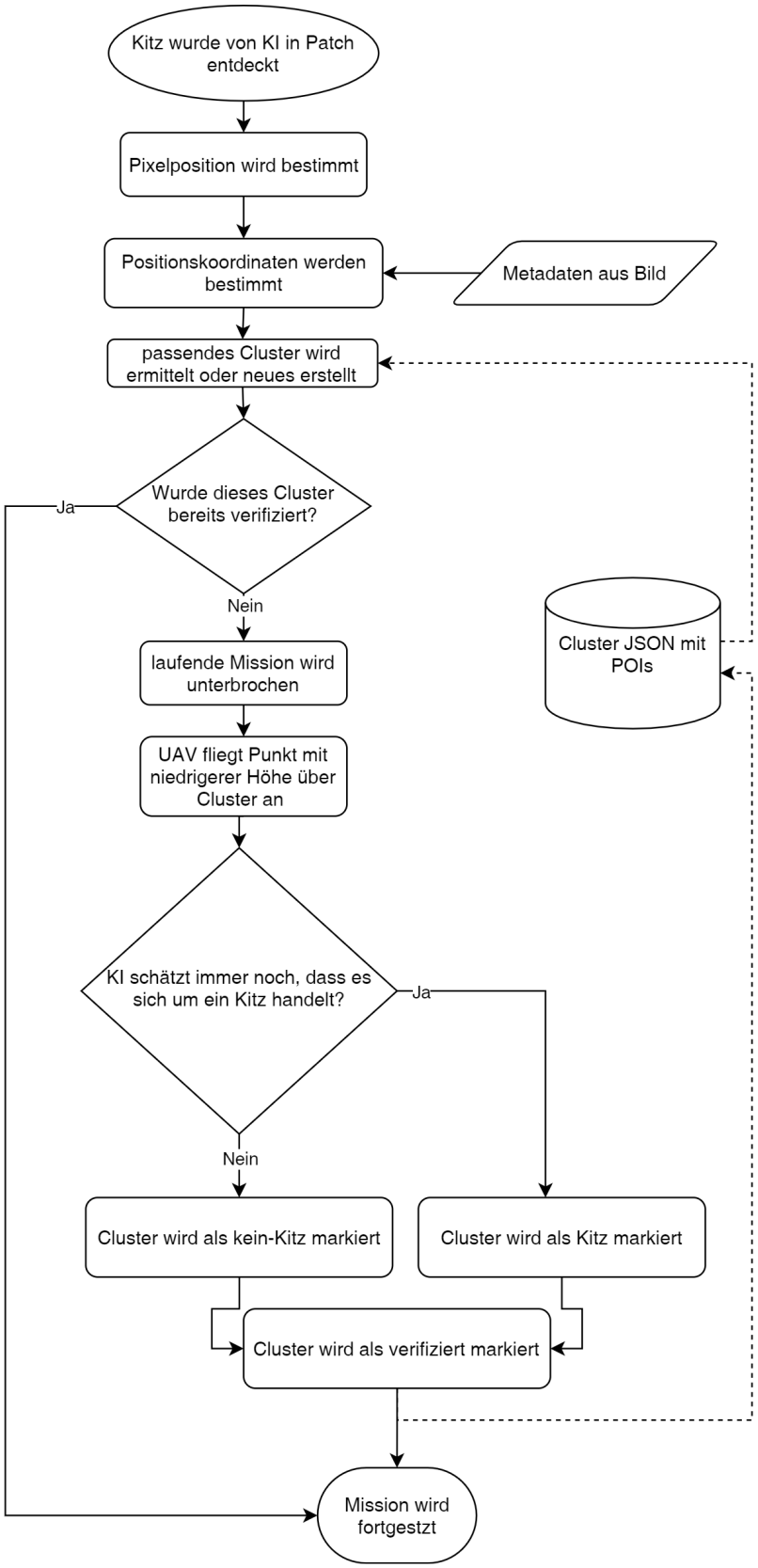

Abbildung 27: Ablauf der Verifikation einer möglichen Kitzfundstelle

Das aktuell für die Verifikation verwendete neuronale Netz ist dasselbe, welches auch für die Entscheidungen aus 40 Metern Flughöhe verwendet wird. Da die KI aber nur für die Bilder aus größerer Entfernung (40 m) konzipiert und trainiert wurde, ist die Performanz aus geringerer Entfernung (5 m) lediglich für Testzwecke ausreichend. Nach Aussage der KI wird hier ein Objekt aktuell meist falsch als Kitz verifiziert, jedoch könnte eine auf die Verifikationshöhe angepasste KI mit dem erhöhten Informationsgehalt vermutlich bessere Resultate erzielen. Da die aufgenommenen Bilder auch während der Verifikation lokal auf einem USB-Massenspeicher abgespeichert werden, wäre eine nachträgliche manuelle Begutachtung und Entscheidung auch zum jetzigen Standpunkt möglich.

4.10 Vergleich der Missionsreihenfolgen

Für die Reihenfolge eines Missionsablaufes gibt es zwei sich in der Verifikation unterscheidende Varianten. Möglichkeit eins ist, dass die Verifikation für alle in einer Aufnahme erkannten Objekte unmittelbar geschieht. Die zweite Variante ist, dass alle zu verifizierenden Signaturen bis zum Ende der Mission gespeichert werden und erst ganz zum Schluss nacheinander abgeflogen werden. Im Folgenden werden beide Methoden gegenübergestellt und deren Vor- und Nachteile verglichen.

- **Unmittelbare Verifikation**: Da bei der unmittelbaren Verifikation, jedes Objekt sofort bei geringerer Höhe angeflogen wird, wird hier eine zusätzliche Zeit für das Auf- und Absteigen benötigt. Bei sehr wenigen Detektionen kann das sofortige Absinken jedoch auch von Vorteil sein, da das UAV zum Ende der Mission möglicherweise weit von der Fundstelle entfernt ist. Für eine manuelle Überprüfung der Verifikation ist die Methode auch von Vorteil, da die Schritte von Erkennung des Objektes bis Verifikation besser auf den Aufnahmen nachvollziehbar sind.

- **Verifikation bei Missionsende**: Hier ist ganz klar die bei vielen Detektionen verkürzte Gesamtmissionszeit von Vorteil. Es sollte hierbei jedoch sichergestellt sein, dass nach Abfliegen der Suchfläche noch ausreichend Akkukapazität für die Verifikation zu Verfügung steht. Um somit einen Akkuwechsel und die damit verbundene Aufwandzeit durch Landung und Start zu vermeiden.

Für eine Abschätzung, ab wie vielen Punkten sich eine nachträgliche Verifikation lohnt, sind einige Randbedingungen festzulegen. Die Zeit, welche das UAV im Mittel für Ab- und Aufsteigen zwischen den Höhen benötigt, wurde zu 40 s gemessen. Bei einer quadratischen beispielhaften Wiese von 5 ha beträgt die

maximale Entfernung zweier Objekte 316 m. Die mittlere Entfernung aller Objekte auf der Wiese ist jedoch stark von der Precision der Erkennungssoftware in den herrschenden Umgebungsbedingungen abhängig, denn je mehr Objekte auf einer Fläche detektiert wurden, desto geringer ist dementsprechend auch der Abstand zwischen diesen. Für die Abschätzung des mittleren Abstandes S_m wird unter Berücksichtigung der Objektanzahl n folgende Formel genutzt:

$$S_m = \frac{316\,\mathrm{m}}{n} \tag{4.3}$$

Die Zeit, welche das UAV vom letzten Missionspunkt bis zum ersten Verifikationspunkt benötigt, wird pauschal auf 30 s festgelegt. Mit der Fluggeschwindigkeit von 5 m/s lässt sich die Verifikationszeit beider Methoden berechnen.
Unmittelbare Verifikation:

$$t_1 = n \cdot 40\,\mathrm{s} \tag{4.4}$$

Verifikation bei Missionsende:

$$t_2 = 40\,\mathrm{s} + 30\,\mathrm{s} + \frac{S_m}{5\,\mathrm{m/s}} \tag{4.5}$$

Hieraus ergibt sich, dass sich die Verifikation bei Missionsende für die Feldfläche von 5 ha zeitlich bereits ab etwa 3 Detektionen lohnt. Um die Zeit bei vielen Detektionen möglichst kurz zu halten, sollte jedoch ein Optimierungsalgorithmus eingesetzt werden. Hier würde sich beispielsweise ein Lösungsalgorithmus für das *Traveling-Salesman*-Problem anbieten [Lin73].

4.11 Test und Einsatz des Gesamtsystems

Um zu überprüfen, ob das Gesamtsystem das erwünschte Verhalten aufweist, wurden mehrere Versuchsflüge durchgeführt. Als aufzuspürende Wärmesignatur wurde das künstliche Kitz (3.3) in einer Wiese mit einer Bewuchshöhe von etwa 50 cm platziert. Der Versuch wurde abends ohne direkte Sonneneinstrahlung und bei einer Umgebungstemperatur von 23 °C verrichtet. Als Flugpfad wurde eine Spur in 40 m Höhe über dem platzierten Objekt gewählt. Die eingestellte Fluggeschwindigkeit betrug 5 m/s. Die Verifikationshöhe lag bei 5 Metern über Grund.

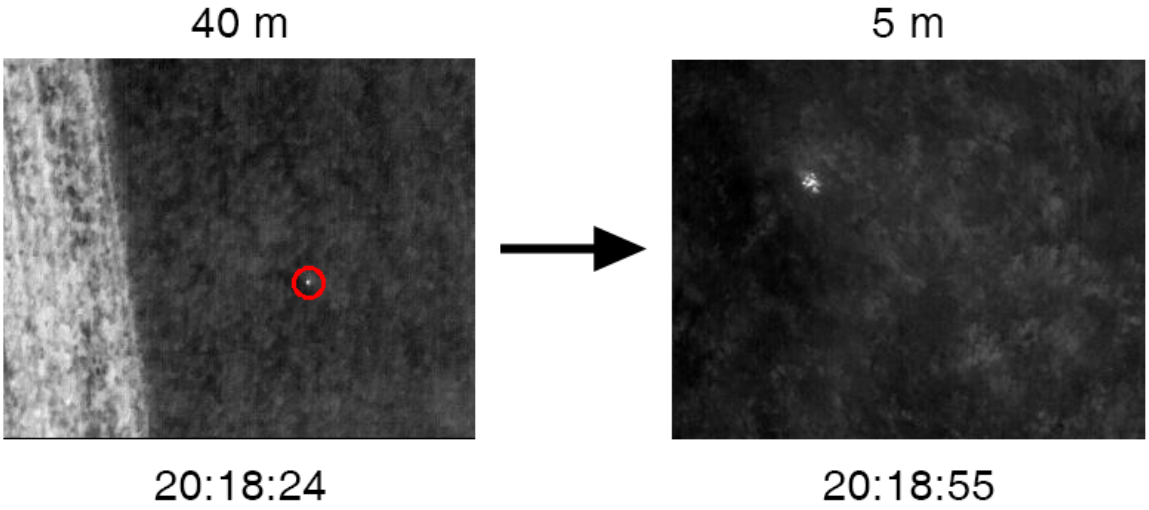

Abbildung 28: Beispielanflug eines POIs

Wie beispielhaft in Abbildung 28 zu sehen, wird der auf Arbeitshöhe erkannte Punkt korrekt angeflogen. Der Abstand vom Bildebenenzentrum zum Objekt wurde auf 80 Pixel gezählt. Dies entspricht bei einer Flughöhe von 5 Metern über Grund einem Abstand von 0,8 Meter.

Das Objekt wurde aufgrund der Bildüberlappung und auch beim Rückflug während des Flugablaufes noch mehrmals erkannt, jedoch nicht angeflogen, was auf eine ordnungsgemäße Funktion des Clusteringalgorithmus rückführbar ist.

5 Messergebnisse

Alle Ergebnisse werden für die kleinere Kamera (320x256 px) ermittelt, da während dieser Arbeit die meisten Versuche mit dieser Kamera verwirklicht wurden. Um einen gewissen Bereich abzusuchen, wird die Wiese in Flugpfade unterteilt (4.1). Die Akkulaufzeit des Trägersystems beträgt im Betrieb etwa 20 Minuten, wodurch sich bei einer durchschnittlichen Fluggeschwindigkeit von 4 m/s eine maximal fliegbare Strecke von 4,8 km ergibt. Diese 4,8 km reichen mit den Flugpfadabständen von 20 m aus, um 7,5 ha Wiese abzusuchen.

5.1 Einfluss von Fehldetektionen auf den maximalen Absuchbereich

In Bayern befinden sich pro Hektar Wiese während der Frühlingsmonate im Schnitt etwa 0,3 Kitze. Also ein Kitz pro 3,5 ha [Isr15]. Im ungünstigsten Fall wurde für die KI eine Precision von 0,23 gemessen (Tabelle 4), jedoch stellte sich im Laufe dieser Arbeit heraus, dass die Precision bei starker Sonneneinstrahlung noch wesentlich weiter sinken kann. Für die Mittagszeit wird nun zur Berechnung eine konservative Precision von 0,1 angenommen. Dies hieße also, dass pro richtig erkanntem Kitz 9 Fehldetektionen auftreten. Im automatisierten Verifikationsbetrieb wird jeder von der KI als Kitz detektierte Punkt, egal ob richtig oder falsch, einmalig angeflogen. In 10 Versuchen wurde die Zeit gemessen, welche das UAV von der Unterbrechung der Mission bis zur Fortsetzung auf Arbeitshöhe benötigt, im Mittel betrug diese Zeit bei einer Verifikationshöhe von 5 m 40 s. Bei 0,3 Kitzen und den daraus resultierenden 2,7 Fehlalarmen pro Hektar ergibt sich also eine zusätzliche Flugzeit von 120 s/ha. Diese muss für die Verifikation beaufschlagt werden. Bei einer Flugzeit pro Hektar von 160 s/ha und der hinzukommenden 120 s/ha für die Verifikation, ergibt sich eine Arbeitszeit von 280 s/ha. Durch das Auf- und Absteigen des Trägersystems wird die Akkulaufzeit wegen dem erhöhten Energieverbrauch um etwa 10 %, also auf 18 Minuten vermindert. Unter diesen erschwerten Bedingungen könnte mit dem System also eine Fläche von maximal 3,8 ha automatisiert abgeflogen werden.

5.2 Abschätzung der Flächenleistung

Für die Abschätzung der Flächenleistung mit Verifikationsprozess ist es zunächst wichtig zu wissen, welche Flächenleistung ohne Verifikationsprozess zu erwarten ist. Im Post Processing Verfahren muss ein Experte die Aufnahmen eines Überfluges im Nachhinein bewerten und sich dabei auf die Bilder aus Arbeitshöhe, also 40 m, verlassen. Die Treffergenauigkeit eines Menschen gegenüber der KI wurde für mittelschwere Umgebungsbedingungen mittels Bildern eines Datensatzes von 2015 gemessen [Isr15], [Sor17]. Unter diesen Bedingungen lag die Precision für den Mensch bei 46 % und für die KI bei 40 %. Somit werden unter den betrachteten Bedingungen vom Menschen 0,65 Alarme pro Hektar angegeben und von der KI 0,75 Alarme pro Hektar.

Da sich das UAV beim Post Processing nicht mehr in der Luft befindet bzw. bereits über dem nächsten Feld, muss eine Person mittels GPS-Tracker die Position aufsuchen und genauer untersuchen. Die durchschnittliche Zeit, welche zum Aufsuchen einer Fundstelle benötigt wird, beträgt 2,5 Minuten [Isr15]. Da jedes auf den Bildern erkannte Objekt angelaufen werden muss, um zu überprüfen, ob es eine Fehldetektion war, beträgt die Verifikationszeit durch eine Person im Mittel 98 s/ha. Zu beachten sind auch die 5 Minuten Auswertzeit, welche pauschal für die Bewertung der Bilder eines Fluges benötigt werden. Bei der maximal mit einer Akkuladung abfliegbaren Fläche von 7,5 ha ergibt sich somit eine hektarbezogene Flugzeit von 298 s/ha. Dies entspricht einer Flächenleistung von 12 ha/h.

Mit automatisierter Detektion und Verifikation dauert für den Fall, dass das Objekt direkt nach der Detektion angeflogen wird, eine Verifikation 40 s. Eine Auswertung im Nachhinein ist im Prinzip nicht notwendig, da die Auswertung bereits von der KI übernommen wird. Bei 0,75 Alarmen pro Hektar wird also mit der Verifikation 30 s/ha verbracht. Die somit ermittelbare hektarbezogene Flugzeit beträgt also 190 s/ha, was einer Flächenleistung von 19 ha/h entspricht.

5.3 Genauigkeit der Fundstellenkoordinaten

Um die Abweichung in der Positionsbestimmung zu ermitteln, wurde in 7 Testflügen der Abstand zwischen der erkannten Wärmesignatur und dem angeflogenen Punkt gemessen. Für die Messung wurde in dem aufgenommenen Bild der Abstand zwischen Bildzentrum und Objekt in Pixeln gezählt und dann anhand der Flughöhe in Meter ungerechnet. Die Ergebnisse sind in folgender Tabelle aufgelistet.

Flug	**Höhe über Grund [m]**	**Abstand [px]**	**berechneter Abstand [m]**
1	5	98	0,98
2	5	146	1,46
3	5	90	0,9
4	5	33	0,33
5	5	95	0,95
6	5	160	1,6
7	40	50	3,8

Tabelle 4: Performanz des verwendeten neuralen Netzes

Die mittlere Genauigkeit beträgt für die Versuchsflüge somit 1,43 Meter. Die große Abweichung bei Flug 7 kommt vermutlich durch eine Desynchronisation zwischen Bildaufnahme und Positionsbestimmung zustande. Genauer untersucht wurde die Ursache jedoch nicht.

6 Fazit und Ausblick

In dieser Arbeit konnte gezeigt werden, dass eine automatisierte Detektion und Verifikation durch On-Board-Bildanalyse, direkter Georeferenzierung und resultierender Flugpfadmanipulation möglich ist. Auch zeigte sich, dass unter bestimmten Bedingungen die automatisierte Rehkitzsuche mit dem entwickelten System effizienter ist als das etablierte Post Processing Verfahren.

Bei erschwerten Bedingungen zeigte das Verfahren jedoch noch starke Schwächen, was hauptsächlich auf die Unzuverlässigkeit des neuronalen Netzes bei direkter Sonneneinstrahlung rückzuführen ist. Hier könnte in zukünftigen Projekten durch optimierte künstliche Intelligenzen eine vermutlich wesentlich bessere Leistung erzielt werden.

Durch die Verifikation und der hierbei entstehenden Bilder aus geringerer Distanz zur gefundenen Wärmesignatur wurde eine genauere Aussage über das detektierte Objekt ermöglicht, ohne dass eine Person zur Fundstelle laufen muss. Durch den gewonnenen Informationsgehalt zur Fundstelle könnte eine hierauf trainierte KI vermutlich zuverlässig und automatisiert verifizieren, ob es sich um ein zu rettendes Rehkitz handelt oder nicht.

Literaturverzeichnis

[Alb14] Alberts, Bruce: *Lehrbuch der Molekularen Zellbiologie*. John Wiley and Sons, 2014 (Wiley-VCH-Lehrbuchkollektion 1). http://onlinelibrary.wiley.com/book/10.1002/3527683267. – ISBN 9783527328246

[Alk17] Alkadhim, Saif A.: *Method to control Pixhawk board using raspberry pi with python*

[Che19] Cheng, Ren: An attempt of building 3D modeling based on UAV tilt photography. In: *Bulletin of Surveying and Mapping* 0 (2019), Nr. 2, S. 161. http://dx.doi.org/10.13474/j.cnki.11-2246.2019.0066. – DOI 10.13474/j.cnki.11–2246.2019.0066

[Con19] Conrady, A. E.: Decentred Lens-Systems. In: *Monthly Notices of the Royal Astronomical Society* 79 (1919), Nr. 5, S. 384–390. http://dx.doi.org/10.1093/mnras/79.5.384. – DOI 10.1093/mnras/79.5.384. – ISSN 0035–8711

[Cra06] Cramer, Michael: *Genauigkeitsuntersuchungen zur GPS/INS-Integration in der Aerophotogrammetrie*, Diss., 2006

[Dua71] Duane, C. B.: Close-range camera calibration. In: *Photogramm. Eng* 37 (1971), Nr. 8, S. 855–866

[Ert16] Ertel, Wolfgang: *Grundkurs Künstliche Intelligenz*. 4. , überarbeitete Auflage. Wiesbaden : Springer Vieweg, 2016 (Computational intelligence). – ISBN 9783658135492 – 9783658135485

[ES09] El-Sheimy, N.: Emerging MEMS IMU and its impact on mapping applications. In: *Photogrammetric week* Bd. 9, 2009

[FLI19] FLIR Systems (Hrsg.): *Flir Boson Datasheet: Compact LWIR Thermal Camera*. www.flir.com. Version: 2019

[Fre18] Freepik Company, S. L. (Hrsg.): *Stem cell diagram on white background*. freepik.com. Version: 2018

[Gol19] Gollapudi, Sunila: *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*. 2019. http://dx.doi.org/10.1007/978-1-4842-4261-2. http://dx.doi.org/10.1007/978-1-4842-4261-2. – ISBN 9781484242612

[Gol20] Goller, Valeria: *Beginn der Mähsaison: Kitze, Hasenkinder & Co in Lebensgefahr – PETA appelliert an Landwirte in Rheinland-Pfalz, Tierbabys zu schützen*. Stuttgart, 2020

[Int85] Internationale Organisation für Normung: *Information Processing ISO-5807*. 1985

[Isr15] Israel, Martin: *Entwicklung eines UAV-basierten Systems zur Rehkitzsuche und Methoden zur Detektion und Georeferenzierung von Rehkitzen in Thermalbildern: Der Fliegende Wildretter*, Diss., 2015. http://dx.doi.org/10.13140/RG.2.2.19289.31841. – DOI 10.13140/RG.2.2.19289.31841

[Kel17] Kellenberger, Benjamin: Fast animal detection in UAV images using convolutional neural networks, 2017

[Kol18] Kolkur, Seema: Convolution Neural Network for Feature Extraction in Skin Disease Detection. In: *Journal of Advanced Research in Applied Artificial Intelligence and Neural network* 5 (2018), Nr. 1&2. https://technology.adrpublications.in/index.php/JofArtificial-Neural-Network/article/view/934

[Läm20] Lämmel, Uwe: *Künstliche Intelligenz: Wissensverarbeitung - Neuronale Netze : mit zahlreichen ... Tabellen*. 5., überarbeitete Auflage. 2020. – ISBN 978–3–446–45914–4

[Lin73] Lin, S.: An Effective Heuristic Algorithm for the Traveling-Salesman Problem. In: *Operations Research* 21 (1973), Nr. 2, S. 498–516. http://dx.doi.org/10.1287/opre.21.2.498. – DOI 10.1287/opre.21.2.498

[Luh10] Luhmann, Thomas: *Nahbereichsphotogrammetrie: Grundlagen, Methoden und Anwendungen*. 3., völlig neu bearbeitete und erweiterte Auflage. Berlin : Wichmann VDE-Verlag, 2010. – ISBN 9783879074792

[Man13] Mancini, Francesco: Using Unmanned Aerial Vehicles (UAV) for High-Resolution Reconstruction of Topography: The Structure from Motion Approach on Coastal Environments. In: *Remote Sensing* 5 (2013), Nr. 12, S. 6880–6898. http://dx.doi.org/10.3390/rs5126880. – DOI 10.3390/rs5126880. – ISSN 2072–4292

[McC43] McCulloch, Warren S.: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Nr. 4, S. 115–133. http://dx.doi.org/10.1007/BF02478259. – DOI 10.1007/BF02478259. – ISSN 1522–9602

[Meh97] Mehrotra, Kishan: *Elements of artificial neural networks*. Cambridge, Mass. : MIT Press, 1997 (A Bradford book). – ISBN 9780262133289

[Mei13] Meier, Lorenz: Mavlink: Micro air vehicle communication protocol. In: *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]* (2013)

[Nex14] Nex, Francesco: UAV for 3D mapping applications: a review. In: *Applied Geomatics* 6 (2014), Nr. 1, S. 1–15. http://dx.doi.org/10.1007/s12518-013-0120-x. – DOI 10.1007/s12518–013–0120–x. – ISSN 1866–928X

[Nex19] Nex, Francesco: Towards Real-Time Building Damage Mapping with Low-Cost UAV Solutions. In: *Remote Sensing* 11 (2019), Nr. 3, S. 287. http://dx.doi.org/10.3390/rs11030287. – DOI 10.3390/rs11030287. – ISSN 2072–4292

[Ngu18] Nguyen, K.: Iris Recognition With Off-the-Shelf CNN Features: A Deep Learning Perspective. In: *IEEE Access* 6 (2018), S. 18848–18855. http://dx.doi.org/10.1109/ACCESS.2017.2784352. – DOI 10.1109/ACCESS.2017.2784352. – ISSN 2169–3536

[Osb06] Osborne, Brian G.: Near-Infrared Spectroscopy in Food Analysis. Version: 2006. http://dx.doi.org/10.1002/9780470027318.a1018. In: *Encyclopedia of Analytical Chemistry*. American Cancer Society, 2006. – DOI 10.1002/9780470027318.a1018. – ISBN 9780470027318

[Pfe12] Pfeifer, Norbert: Direct georeferencing with on board navigation components of light weight UAV platforms. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39 (2012), Nr. B7, S. 487–492

[Qia20] Qiang, Fang: Defects detection in infrared thermography by deep learning algorithm. In: *Thermosense: Thermal Infrared Applications XLII* Bd. 11409, 2020, S. 114090T

[Raz14] Razavian, Ali: CNN Features Off-the-Shelf: An Astounding Baseline for Recognition, 2014

[Rog87] Rogers, Lynn L.: Rectal temperatures of 2 free-ranging white-tailed deer fawns. In: *Journal of Wildlife Management* 51 (1987), S. 59–62

[Sch09] Schmidts, Rudi: *Basiswissen Videoproduktion: Verzeichnung*. www.slashcam.de/artikel/Basiswissen-Videoproduktion/Verzeichnung.html. Version: 2009

[Sch18] Scholz, Mathias: *Die Physik der Sterne: Aufbau, Entwicklung und Eigenschaften*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2018. http://dx.doi.org/10.1007/978-3-662-57801-8. http://dx.doi.org/10.1007/978-3-662-57801-8. – ISBN 978–3–662–57801–8

[Sha17] Sharma, Sagar ; Towards Data Science (Hrsg.): *Activation Functions in Neural Networks: Sigmoid, tanh, Softmax, ReLU, Leaky ReLU explained*. towardsdatascience.com. Version: 2017

[Sor17] Sorg, Michael: *Konzeption und Implementierung eines neuronalen Netzes zur Detektion von möglichen Rehkitzen anhand von Thermalbildern*, Hochschule Baden-Württemberg Mannheim, BACHELORARBEIT, 2017

[Ste05] Stewart, M.: Infrared thermography as a non-invasive tool to study animal welfare. In: *Animal Welfare* 14 (2005)

[Sze06] Szeliski, Richard: Image Alignment and Stitching: A Tutorial. In: *Found. Trends. Comput. Graph. Vis.* 2 (2006), Nr. 1, S. 1–104. http://dx.doi.org/10.1561/0600000009. – DOI 10.1561/0600000009. – ISSN 1572–2740

[Tor18] Toro, Felipe G.: *UAV sensors for environmental monitoring*. First edition. Basel and Beijing and Wuhan and Barcelona and Belgrade : MDPI, 2018. – ISBN 9783038427544

[Usa14] Usamentiaga, Rubén: Infrared Thermography for Temperature Measurement and Non-Destructive Testing. In: *Sensors* 14 (2014), Nr. 7, 12305–12348. http://dx.doi.org/10.3390/s140712305. – DOI 10.3390/s140712305. – ISSN 1424–8220

[Vil08] Villiers, Jason P.: Centi-pixel accurate real-time inverse distortion correction. In: *Optomechatronic Technologies 2008* Bd. 7266, 2008, S. 726611

[Wal20] Walker, Samantha: Low-altitude aerial thermography for the archaeological investigation of arctic landscapes. In: *Journal of Archaeological Science* 117 (2020), S. 105126

[Wan20] Wang, Dashuai: UAV environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution. In: *Computers and Electronics in Agriculture* 175 (2020), 105523. http://dx.doi.org/10.1016/j.compag.2020.105523. – DOI 10.1016/j.compag.2020.105523. – ISSN 0168–1699

[Wil18] Willee, Hamish: *Mavlink Devguide*. 2018

[Wu20] Wu, Jianxin: *Convolutional neural networks*. https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf. Version: 2020

[Zab20] Zabel, Frank: Jungwildrettung: Einstieg leicht gemacht. In: *Pirsch* (2020), Nr. 07

[Zha19] Zhang, Chi: Data-driven ship energy efficiency analysis and optimization model for route planning in ice-covered Arctic waters. In: *Ocean Engineering* 186 (2019), 106071. http://dx.doi.org/10.1016/j.oceaneng.2019.05.053. – DOI 10.1016/j.oceaneng.2019.05.053. – ISSN 0029–8018

Abbildungsverzeichnis

Tabellenverzeichnis