

Westfälische Wilhelms-Universität Münster

Institut für Informatik

Praktikum *Computer Vision: Camera Trap Challenge* – Wintersemester 2018/19

Dozenten: Jun.-Prof. Dr. Benjamin Risse, Andreas Nienkötter

Auswertung von Kamerafallenbildern mithilfe von Principal Components Analysis, Spatial Pyramid Matching und Support Vector Machines

Thomas Poschadel
Rudolf-Harbig-Weg 36, 48149 Münster
M.Sc. Informatik
Matrikelnummer: 123 456
blabla@wwu.de

Joschka Strüber
Rudolf-Harbig-Weg 36, 48149 Münster
M.Sc. Informatik
Matrikelnummer: 418 702
j.st@wwu.de

Sufian Zaabalawi
Straße Hausnummer, 12345 Münster
M.Sc. Informatik
Matrikelnummer: 123 456
blabla@wwu.de

Münster, 27. März 2019

Inhaltsverzeichnis

1 Einleitung	1
2 Aufteilung auf Sequenzen	2
2.1 Sortierung mithilfe der Exif-Daten	2
2.2 Camera Trap Sequencer	3
3 Lokalisierung mit Principal Components Analysis (PCA)	4
3.1 Hintergrundapproximation mit PCA	4
3.2 Sliding-Window Lokalisierung mit PCA	7
3.3 Objektdetektion mit PCA	7
3.3.1 Algorithmus	7
4 Klassifizierung mit Histograms of Gradients und Support Vector Machines	9
5 Klassifizierung mit Spatial Pyramid Matching	9
5.1 Grundidee Spatial Pyramid Matching	9
5.2 Locality-constrained Linear Coding	9
5.3 Implementierung	9
6 Evaluierung	9
6.1 Daten	9
6.2 Laufzeit Spatial Pyramid Matching	9
6.3 Güte der Klassifizierungstechniken	10
7 Fazit	10
Literaturverzeichnis	11
Eigenständigkeitserklärung	11

1 Einleitung

Kamerafallen bieten eine immer wichtiger werdende Möglichkeiten Populationen zu überwachen und erforschen. Forschungsinteressen sind beispielsweise die Veränderung der Biodiversität, der Einfluss des Klimawandels und anderer Einflüsse auf die Lebensräume und die Migrationsmuster von Populationen [Yu et al. 13].

Durch die immer größer werdende Akzeptanz von Kamerafallen, steigende Qualität und sinkende Preise kommt es zu einer exponentiell wachsenden Datenmenge. Diesen Daten manuell Herr zu werden stellt eine Herausforderung dar, denn jedes Bild muss einzeln von einem Experten auf Tiere untersucht werden. Aufgrund der Verwendung von unveröffentlichten Daten in aktuellen Forschungsprojekten ist Crowd-Sourcing oft unmöglich. Zudem zeichnen sich die anfallenden Bilder durch einen hohen Anteil von False-Positives (Bilder ohne Tiere) und eine große Vielfalt von Arten in verschiedensten Posen, Entferungen und bei wechselnden Witterungsbedingungen aus, was die automatische Analyse erschwert.

Im Kontext dieser Arbeit beziehen wir uns dabei auf einen Datensatz aus dem niederländischen Nationalpark De Hoge Veluwe. Die Datenbank umfasst 40 GB Bilder von neun einheimischen Tierspezies, die mithilfe von Reconyx-Kamerafallen gesammelt wurden. Dabei wurden sowohl Farbbilder am Tag als auch Infrarotbilder in schwarz-weiß in der Nacht aufgenommen.

Um diese komplexe Aufgabe zu automatisieren, stellen wir eine Softwarepipeline vor, mit deren Hilfe es möglich ist alle nacheinander anfallenden Herausforderungen zu lösen. Der erste Schritt besteht in der Ordnung der Daten. Dafür haben wir mit dem *Camera Trap Sequencer* eine Software mit grafischer Benutzeroberfläche implementiert, die es dem Benutzer erlaubt nach Tierarten vorsortierte Datenbanken oder einzelne Ordner von Bildern auf zusammenhängende Sequenzen aufzuteilen.

Der nächste Schritt ist die Lokalisierung von Tierarten in Bildern. Das ermöglicht zum einen das Aussortieren von Bildern, die in Wirklichkeit keine Tiere zeigen, und zum anderen die Identifikation von Bildausschnitten, die für die spätere Klassifizierung relevant sind. Hierfür verwenden wir eine Pipeline mit verschiedenen Vor- und Nachbereitungsschritten, die mithilfe von *Principal Components Analysis* ein Hintergrundbild auf einer Bildsequenz berechnet und somit die Segmentierung von relevanten Bildausschnitten erlaubt. Um auch die Lokalisierung von Tieren auf Einzelbildern zu ermöglichen, wurde zusätzlich ein PCA-unterstütztes *Sliding-Window*-Verfahren implementiert.

Den Abschluss jeder Auswertung bildet das Klassifizieren von Spezies in zuvor bestimmten *Regions of Interest*. Hierzu stellen wir zwei verschiedene Techniken vor:

Die erste ist die Klassifizierung mit Hilfe einer *Support Vector Machine* mit *Radial-Basis-Function*-Kernel auf dem *Histogram-of-oriented-Gradients*-Feature, einem Strukturfeature. Das zweite Verfahren ist *Spatial Pyramid Matching* (SPM) mit *Locality-constrained linear Coding* [LSP06]. Hierbei wird das Eingabebild in immer feinere Teilbilder unterteilt, auf denen dann SIFT- oder

LBP-Features berechnet werden. Diese Features werden mit LLC kodiert, wobei die räumliche Aufteilung erhalten bleibt. Abschließend werden die so bestimmten Codes zum Trainieren einer Support Vector Machine mit linearem Kernel benutzt, da sich empirisch erwiesen hat, dass sie gut linear separierbar sind [Yang et al. 09].

Zum Abschluss unserer Ausarbeitung evaluieren wir unsere Verfahren. Betrachtet werden sowohl die Laufzeit der Algorithmen als auch ihre Güte auf den uns zur Verfügung stehenden Daten. Da der ursprüngliche Datensatz zu groß ist, betrachten wir dabei lediglich zwei repräsentative Teilmengen: In der DDD befinden sich Tagbilder von Dachsen und Damhirschen. Die geringe Datenmenge erlaubte schnelle Ergebnisse, ohne grundlegende Probleme, wie beispielsweise unterschiedlich unbalancierte Klassenhäufigkeiten, aus dem Blick zu verlieren. Für die DDD+ haben wir über 2000 Tag- und Nachtbilder von insgesamt sechs verschiedenen Tierarten zusammengestellt. Ziel ist hierbei die Bestimmung der Güte des Verfahrens auf einem komplexen Datensatz.

2 Aufteilung auf Sequenzen

2.1 Sortierung mithilfe der Exif-Daten

Die handelsüblichen Kamerafalle haben einen Sensor, der bei Bewegung auslöst und zunächst zehn Bilder im Abstand von jeweils einer Sekunde schießt. Sollte es am Ende einer Zehnersequenz weiterhin Bewegung im Sichtfeld der Kamera geben, löst der Sensor erneut aus und es werden weitere zehn Bilder aufgenommen. Das sorgt dafür, dass der Datensatz aus zusammenhängenden Sequenzen von jeweils zehn oder mehr Bildern besteht. Oft befinden sich gerade auf den letzten Aufnahmen einer Sequenz keine Tiere mehr, da sich diese aus dem Bild bewegt haben.

Unglücklicherweise sind die Daten auf der Datenbank lediglich nach Tierart sowie dort jeweils nach Tag und Nacht, leeren Bildern und Fehlklassifizierungen sortiert. Für das korrekte Funktionieren unserer Segmentierungstechnik PCA ist es aber nötig, dass die Daten in zusammenhängenden Sequenzen vorliegen. Aus diesem Grund benutzen wir die Exif-Metadaten, um die Daten aufzuteilen. Diese Exif-Daten umfassen Informationen über das Bild, wie beispielsweise die Abmessungen, das Aufnahmedatum, die Belichtungszeit, den ISO-Wert und das Kameramodell.

Unser Algorithmus sammelt zunächst die Metadaten aller aufzuteilenden Bilder in einer Liste. Diese Liste wird primär nach Seriennummer der Kamera und sekundär nach Aufnahmezeitpunkt sortiert. Sollte es zwischen zwei benachbarten Bildern in der Liste zu einem Wechsel der Seriennummer kommen, so wissen wir, dass eine neue Sequenz beginnt. Ebenso gilt das für zwei Bilder, die von derselben Kamera aufgenommen wurden, deren Aufnahmezeitpunkte sich jedoch um mehr als ein paar Sekunden unterscheiden. Diese beiden Situationen markieren den Wechsel einer Sequenz anhand der wir unterscheiden können, welche Bilder zusammenhängen.

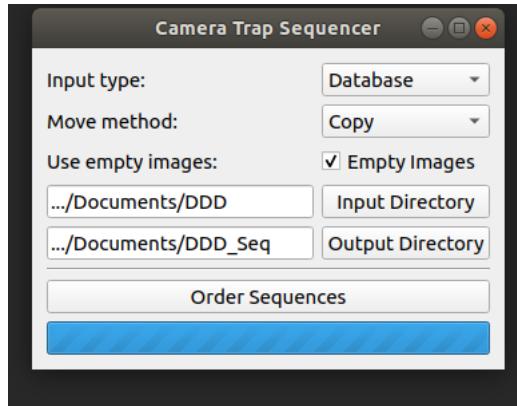


Abbildung 1: Der Camera Trap Sequencer im Linux-Design.

Leider gehört die Seriennummer nicht zu den ursprünglichen Exif-Metadaten. Stattdessen wird sie in die sogenannten *Maker Notes* geschrieben, einem freien Datenfeld, dass die Kamerahersteller für ihre Zwecke benutzen können. Aktuell gibt es keine Python-Bibliothek, die das Maker-Note-Feld auslesen kann. Aus diesem Grund waren wir gezwungen die Perl-Bibliothek „ExifTool“ [PH03] zu benutzen und auf sie mit Hilfe einer Python-Schnittstelle zuzugreifen.

Zugegebenermaßen ist die Verwendung der Metadaten nicht der eleganteste Weg Sequenzen zu bestimmen. Sie hat aber - anders als das Auslesen und Abgleichen der Pixel aus der linken oberen Ecke des Bildes - den Vorteil unabhängig vom Kamerahersteller und -modell zu sein und ist sehr effizient.

2.2 Camera Trap Sequencer

Der Camera Trap Sequencer ermöglicht es dem Anwender seine Datenbank mit Kamerafallenbildern unkompliziert und schnell auf Sequenzen aufzuteilen. Als Eingabe kann sowohl eine komplette Datenbank als auch ein einzelner Ordner mit Bildern benutzt werden. Man kann sich zwischen dem Verschieben und Kopieren der Bilder entscheiden. Das Verschieben von Bildern hat den Vorteil, dass keine Daten dupliziert werden. Die ursprüngliche Ordnerstruktur bleibt momentan jedoch noch erhalten.

Abschließend hat man im Fall einer Bilddatenbank die Möglichkeit sich dafür zu entscheiden mit Informationen über leere Bilder zu speichern. Alle vorsortierten Datenbanken haben Verzeichnisse mit dem Namen „empty“, in denen sich Bilder auf Sequenzen befinden, auf denen keine Tiere zu sehen sind. Unsere Segmentierungstechnik PCA ist darauf angewiesen möglichst auf möglichst langen Bildsequenzen angewendet zu werden. Deshalb ist es nützlich, False-Positives mitzuverwenden, insbesondere weil leere Bilder einen großen Beitrag zur Berechnung eines leeren Hintergrundbilds leisten können. Da wir als Label für die Klassifizierung aber die Namen der Tierordner benutzen, müssen diese vor der Klassifizierung aussortiert werden. Deshalb speichern wir, falls die empty-Option gesetzt ist, eine Textdatei mit den Dateipfaden aller leeren Bilder, die dann im Anschluss an PCA aussortiert werden können.

Der Camera Trap Sequencer selbst ist mit PyQt5 umgesetzt. Er zeichnet sich deshalb durch ein natives Design auf jeder Plattform aus.

3 Lokalisierung mit Principal Components Analysis (PCA)

Die automatische Lokalisierung von Objekten in digitalen Bildern ist ein wesentlicher Bestandteil vieler Anwendungen. Für das Lokalisierungsproblem in dieser Arbeit bietet sich die Verwendung von der Methoden *Hintergrund-Subtraktion* und *Sliding-Window* mit PCA an.

3.1 Hintergrundapproximation mit PCA

Um Bewegungen in Bildsequenzen erkennen zu können, wird in der Praxis sehr häufig das Verfahren der *Hintergrund-Subtraktion* angewandt. Dabei handelt es sich um ein klassisches Verfahren aus dem Bereich der Bilderkennung. Das Hintergrundbild kann mithilfe von PCA approximiert werden. Anschließend wird das Vordergrundbild über die Differenz zum Hintergrundbild extrahiert. PCA, oder auch Hauptkomponentenanalyse, ist ein statistisches Verfahren um große Mengen von Datensätzen zu vereinfachen und zu strukturieren, indem die Datenpunkte im p -dimensionalen Raum R^p in einen q -dimensionalen Unterraum R^q mit ($q < p$) projiziert werden. Diese Transformation muss dabei so gewählt werden, dass möglichst wenig Information verloren geht. Grundsätzlich benutzt PCA die *Niedrigrangapproximation*. Damit kann eine Matrix durch eine andere Matrix im allgemeinen Rang angenähert werden. Sei eine Matrix A mit $\text{Rang}(A) = r$ und $r > k$:

$$\min_{\text{rang}(A)=k} \|A - B\|_2 \quad (1)$$

Dabei soll die Differenz zwischen A und B minimiert werden. Mithilfe der *Singulärwertzerlegung* (SVD) können die Singulärwerte einer Matrix abgelesen werden. Die SVD von Matrix A ist dann:

$$A = U\Sigma V^T \quad (2)$$

Somit kann ein Hintergrundbild aus einer Sequenz von Bildern wie folgt approximiert werden (Abbildung 2):

- ▷ Berechne Singulärwertzerlegung aller Bildern von Sequenz X :

$$SVD(X) = C = U\Sigma V^T \quad (3)$$

- ▷ Leite die Matrix Σ_k von Σ her, sodass die Werte $n - k$ entlang der Diagonale durch 0 ersetzt werden.

- ▷ Dies ergibt die Niedrigrangapproximation von Matrix X :

$$SVD(X)_k = C_k = U\Sigma_k V^T \quad \text{mit} \quad \Sigma_k = \text{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0) \quad (4)$$

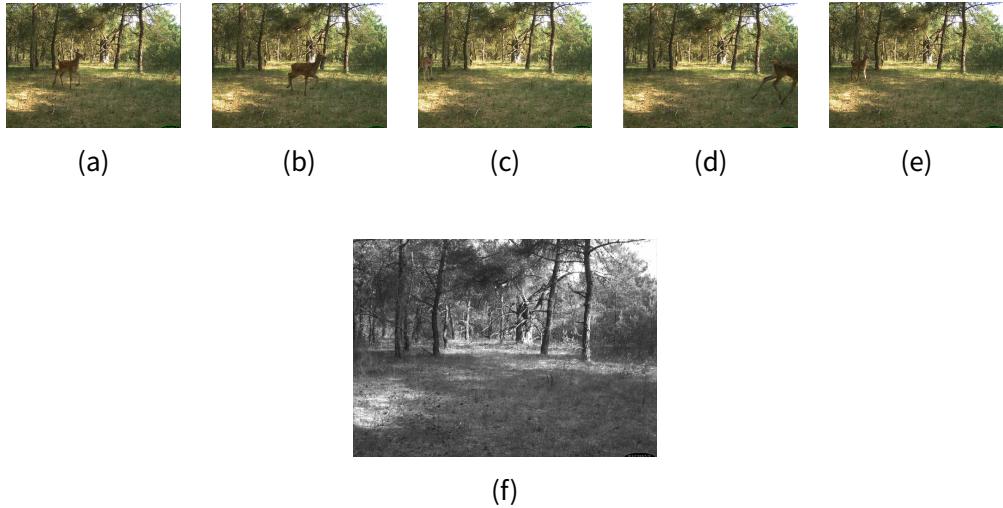


Abbildung 2: (a)-(e) Bilder aus einer Sequenz und (f) das approximierte Hintergrundbild.

Anschließend kann das Vordergrundbild durch die klassische *Hintergrund-Subtraktion* extrahiert werden (Abbildung 3).

$$\begin{array}{ccc} \text{M} & - & \text{L} = \text{S} \end{array}$$

The diagram illustrates the subtraction process. On the left is a color image of the deer in the forest. In the center is a grayscale version of the same image, labeled L. To the right is a grayscale image showing only the deer, labeled S. Between the first two images is a minus sign, and between the second and third is an equals sign, indicating the subtraction operation.

Abbildung 3: Das Vordergrundbild ergibt sich durch die Subtraktion des approximierten Hintergrundbildes.

Zum Nachbearbeitung des Vordergrundes gehört eine Vielzahl von Operationen z.B. *morphologische Operationen*, *Thresholding* und *Filterung*. Damit können kleinere Bildstrukturen und Rauschen entfernt, vergrößert, geschlossen oder aufgefüllt werden. Können jedoch diese Operationen zu einer Veränderung der Größe der Vordergrundelemente führen, was zur Lokalisierung des Elements aber keinen Störfaktor ergibt. Durch Kombination der Operationen in einer bestimmten Reihenfolge kann Größenveränderung verhindert und dennoch die Vorteile der Operationen genutzt werden. Durch *Opening* werden zunächst kleine Strukturen bzw. Rauschen, welches zum Hintergrund gehört, entfernt. Danach werden kleine Löcher innerhalb der Vordergrundelemente durch *Closing* geschlossen. In (Abbildung 4) ist eine Kombination dieser Pipeline zur Nachbearbeitung des Vordergrundes benutzt worden.

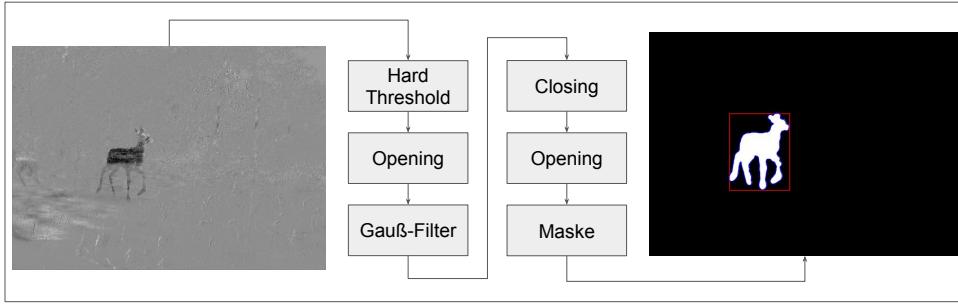


Abbildung 4: Die Pipeline der Nachbearbeitung des Vordergrundbildes. Durch *Opening* und *Closing* werden kleine Bildstrukturen bzw. Rauschen entfernt und kleine Löcher geschlossen werden. Die Gauß-Filterung dient in diesem Fall dazu, die Silhouette des Vordergrundelements grob zu vergrößern.

3.2 Sliding-Window Lokalisierung mit PCA

Sliding-Window ist eine Brute-Force-Suche über das Bild mit fester Fenstergröße, um Objekte zu finden. Für jedes dieser Fenster wird ein Bildklassifikator angewendet, um zu bestimmen, ob das Fenster ein bekanntes Objekt enthält. In diesem Fall wird PCA als Objekt-Klassifikator angewandt.

3.3 Objektdetection mit PCA

Jedes Bild ist ein Punkt in einem hochdimensionalen Raum. Durch das PCA-Verfahren lassen sich die Datenpunkte in einen niederdimensionalen Unterraum abbilden. PCA sucht die ersten k -Hauptkomponenten, welche die Daten mit einer maximalen Varianz beschreiben. Damit wird es eine niederdimensionale Darstellung gefunden, bei der die Klassifizierung leichter wird.

3.3.1 Algorithmus

- ▷ Phase I: Initialisierung
 - ▷ Berechne das Mittelwertbild der Trainingsbilder

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad (5)$$

- ▷ Berechne die zentrierten Daten durch Subtraktion der Trainingsbilder vom Mittelwertbild

$$C = X - \mu \quad (6)$$

- ▷ Berechne die Eigenwerte und Eigenvektoren für die Kovarianzmatrix CC^T

$$\text{SVD}(C) = \mathbf{U}\Sigma\mathbf{V}^T \quad (7)$$

- ▷ Projiziere die Trainingsbilder in den r -Unterraum

$$\mathbf{Y} = \mathbf{U}_r^T C \quad (8)$$

- ▷ Phase II: Klassifikation

Gegeben ist ein unbekanntes Bild M

- ▷ Projiziere das Bild M in den r -Unterraum

$$\mathbf{W} = \mathbf{U}_r^T (M - \mu) \quad (9)$$

- ▷ Finde den nächsten Nachbarn zwischen den projizierten Trainingsbildern \mathbf{Y} und dem projizierten Bild \mathbf{W} .

Die Sliding-Windows laufen das Bild mit unterschiedlichen Fenstergrößen durch. Demnach werden Schnittbilder einzelne mit PCA klassifiziert. Dabei wird der nächste Nachbar der projizierten Schnittbilder gefunden und zugeordnet (Abbildung 5).

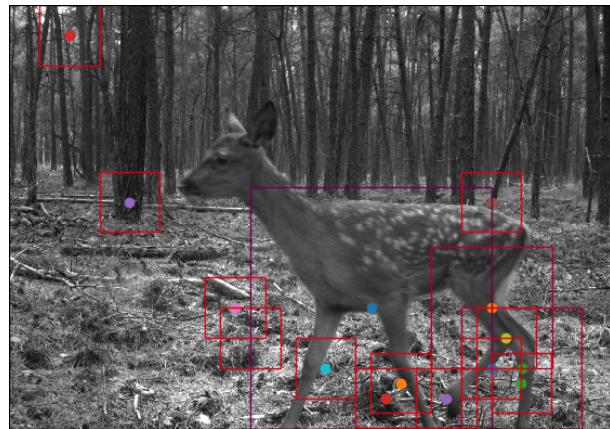


Abbildung 5: Lokalisierung mit Sliding-windows und PCA.

4 Klassifizierung mit Histograms of Gradients und Support Vector Machines

5 Klassifizierung mit Spatial Pyramid Matching

5.1 Grundidee Spatial Pyramid Matching

-hintergrund: Lazebnik et al., Scene Classification -aufteilung in immer feinere regionen -bild mit dense features und spatial pyramid einfügen -features für alle regionen (sift, lbp) -berechnungen von codes aufgrund dieser features (enkodierung, pooling, normalisierung) -trainieren einer SVM mit den codes

5.2 Locality-constrained Linear Coding

-kurze einleitung -vortraining des codebooks mit clustering -vergleich mit vector quantization -gleichung -lösungsalgorithmus -pooling -normalisierung -lineare separararkeit, laufzeit, bezug auf hog kapitel -kurze erwähnung der svm mit squared-hinge-loss und

5.3 Implementierung

-umsetzung numpy, optimierung numba -opencv für sift, scikit-image für lbp -clustering und linear svm mit scikit-learn -transformer mit scikit-learn, einbettung in pipeline

6 Evaluierung

-hinweis auf laufzeit und klassifizierung -pca nicht testbar, da keine ground-truth bilder

6.1 Daten

-hoge veluwe -kamerafallenbilder, sequenzen -größe -spezies -DDD/DDD+

6.2 Laufzeit Spatial Pyramid Matching

-sequentiell -multiprocessing -numba -hauptproblem:lösung des gleichungssystems -ausblick: umsetzung mit tensorflow

6.3 Güte der Klassifizierungstechniken

-HOG: auf DDD und DDD+ -SPM: -random search cv mit fünf folds -scipy reciprocal -versch. Größen
-SIFT Güte auf DDD -LBP Güte auf DDD -SIFT Güte auf DDD+ -kombination von SIFT und LBP auf DDD+

7 Fazit

-kurzbeschreibung der verfahren und ergebnisse -ausblick: -tensorflow: schnellere laufzeit -> training mit größeren codebooks auf mehr daten -ensemblemethoden zur besseren kombination von sift und clbp: soft voting (SVC, statt LinearSVC), boosting

Literatur

- [LSP06] Svetlana Lazebnik, Cordelia Schmid u. a. „Beyond bags of features: spatial pyramid matching for recognizing natural scene categories“. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2006.
- [PH03] *ExifTool* by Phil Harvey. <https://www.sno.phy.queensu.ca/~phil/exiftool/>. Accessed: 04.03.2019.
- [Yang et al. 09] Jianchao Yang, Kai Yu u. a. „Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification“. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.
- [Yu et al. 13] Xiaoyuan Yu, Jiangping Wang u. a. „Automated identification of animal species in camera trap images“. In: *EURASIP Journal on Image and Video Processing* (2013).

Eigenständigkeitserklärung

Hiermit versichern wir, dass die vorliegende Ausarbeitung *Auswertung von Kamerafallenbildern mit Hilfe von Principal Components Analysis, Spatial Pyramid Matching und Support Vector Machines* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

(Ort, Datum)

(Unterschrift)