

# Camera Trap Challenge

Praktikum Computer Vision

# Aufbau

1. Aufteilung auf Sequenzen
2. Segmentierung und Lokalisierung
3. Lokalisierung und Klassifizierung mit HOGs und SVMs
4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs
5. Evaluierung
6. Ausblick

## 1. Aufteilung auf Sequenzen

## 2. Segmentierung und Lokalisierung

## 3. Lokalisierung und Klassifizierung mit HOGs und SVMs

## 4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs

## 5. Evaluierung

## 6. Ausblick

## Aufteilung auf Sequenzen

- ▶ sortiere Bilder nach Seriennummer der Kamera und Zeitpunkt der Aufnahme
- ▶ dafür notwendig: Zugriff auf die EXIF Maker Notes mit Hilfe des Programms „Exiftool“ [PH03]
- ▶ unterteile die sortierte Folge von Bildern in Sequenzen, falls sich die Seriennummer ändert oder der Zeitunterschied zwischen zwei Bildern zu groß wird
- ▶ Umsetzung in Programm „Camera Trap Sequencer“:
  - ▶ setzt viele nützliche Funktionen zum Aufteilen auf Sequenzen um
  - ▶ GUI implementiert mit Qt 5

# Camera Trap Sequencer

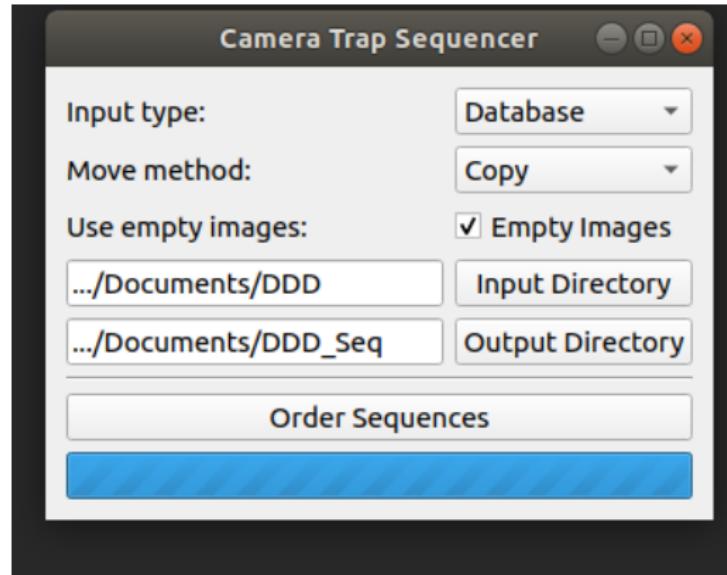


Abbildung: Grafische Benutzeroberfläche zur Aufteilung von Kamerafallensequenzen.

1. Aufteilung auf Sequenzen

2. Segmentierung und Lokalisierung

3. Lokalisierung und Klassifizierung mit HOGs und SVMs

4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs

5. Evaluierung

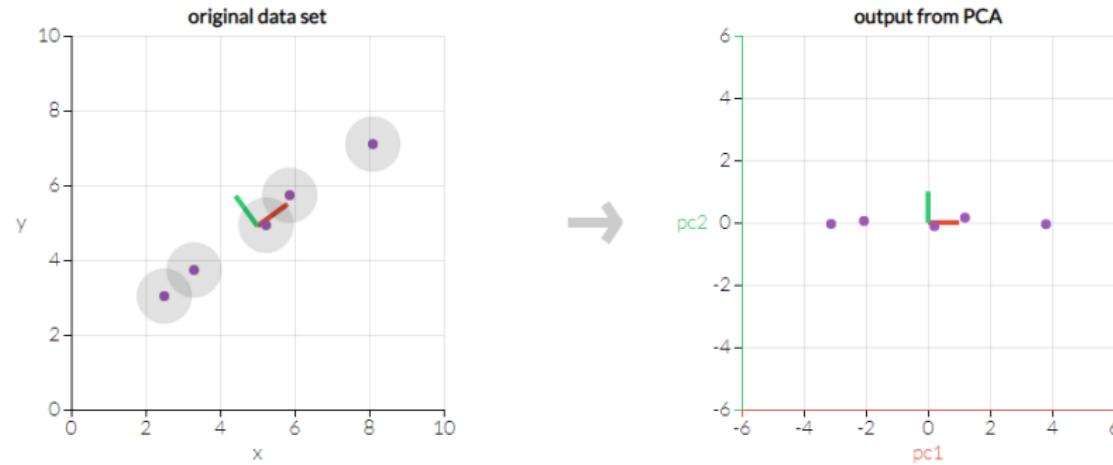
6. Ausblick

## Bewegungsdetektion mit Principal Component Analysis (PCA)

- ▶ Bewegungsdetektion durch Vordergrund- und Hintergrunderkennung
- ▶ Vorgehen:
  - ▶ betrachte Datensatz als Sequenz von Bildern
  - ▶ berechne redundante Informationen (Hintergrund)
  - ▶ Den Vordergrund erhält man, indem man das Low-Rank Hintergrundbild vom Datensatz subtrahiert.

# Principal Component Analysis (PCA)

- ▶ lineare Transformation der Variablen
- ▶ Projizierung in einen neuen Unterraum
- ▶ Dimensionen reduzieren



<http://setosa.io/ev/principal-component-analysis/>

## Bewegungsdetektion mit PCA

- ▶ Vordergrund- und Hintergrunderkennung
- ▶ Berechnung vom Vordergrund:
  - ▶ PCA sucht die ersten  $k$ -Hauptkomponenten, die die Daten mit maximaler Varianz beschreiben.
  - ▶ Hierbei ist  $M$  das Eingabebild und  $L$  ist die redundante Bildinformation mit Rang- $k$ .

# Bewegungsdetektion mit PCA

- ▶ Low-Rank Approximation
  - ▶ Berechne Singulärwertzerlegung aller Bildern im Sequenz X

$$\text{SVD}(X) = C = U\Sigma V^T \quad (1)$$

- ▶ Leite die Matrix  $\Sigma_k$  von  $\Sigma$ , sodass die  $n - k$  Werte entlang der Diagonale durch 0 ersetzt werden.
- ▶ Ergibt die Low-Rank Approximation

$$C_k = U\Sigma_k V^T \quad (2)$$

## Bewegungsdetektion mit PCA

- ▶ Vordergrund- und Hintergrunderkennung
  - ▶ Berechnung vom Hintergrund:

$$L = C_1 = U\Sigma_1V^T \quad (3)$$

Originalbild



M

Hintergrund



L

## Bewegungsdetektion mit PCA

- ▶ Vordergrund- und Hintergrunderkennung
  - ▶ Berechnung vom Vordergrund:



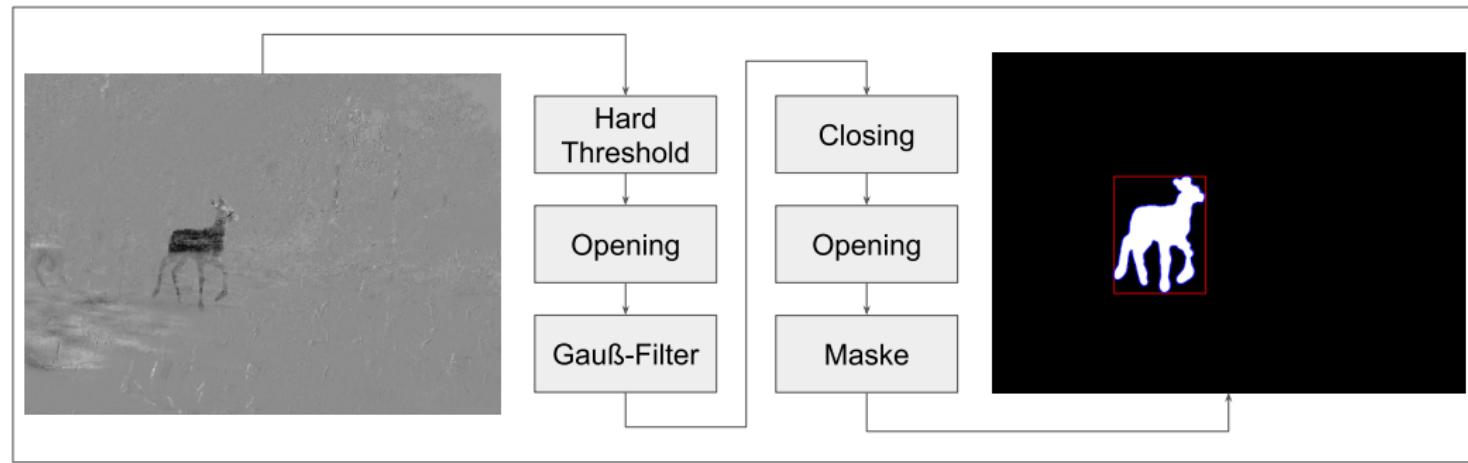
$$M - L = S$$

## Bewegungsdetektion mit PCA

- ▶ Nachbearbeitung vom Vordergrund
  - ▶ Hard- und Soft Thresholding
  - ▶ Aktive Konturen
  - ▶ Median- und Gauß-Filterung
  - ▶ Morphologische Operationen

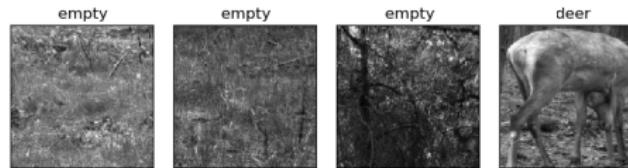
## Bewegungsdetektion mit PCA

- ▶ Nachbearbeitung vom Vordergrund:



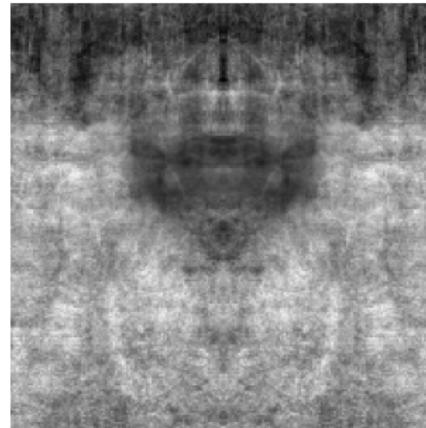
# Objektdetektion mit PCA

- ▶ Vorbearbeitung der Daten:



# Objektdetektion mit PCA

## 1. Daten Zentrieren



Mittelwert Bild

$$C = X - \bar{x}$$

(4)

## Objektdetektion mit PCA

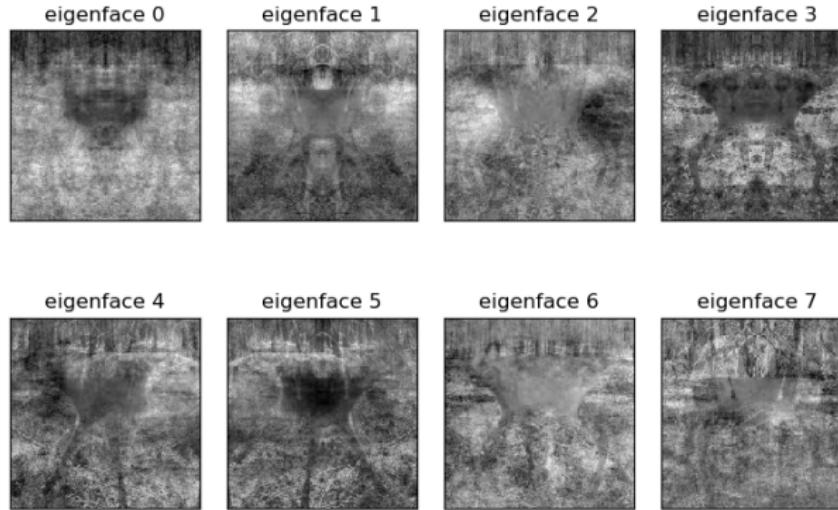
2. Berechne die Eigenwerte und die Eigenvektoren für die Kovarianzmatrix  $CC^T$ :

$$\text{SVD}(C) = U\Sigma V^T \quad (5)$$

3. Projektion der Datensatz  $X$  in den  $r$ -Unterraum:

$$Y = U_r^T(X - \bar{x}) \quad (6)$$

## Objektdetektion mit PCA



# Objektdetektion mit PCA

predicted: badger  
true: badger



predicted: badger  
true: badger



predicted: deer  
true: deer



predicted: deer  
true: deer



predicted: deer  
true: deer



predicted: empty  
true: empty



predicted: deer  
true: deer



predicted: deer  
true: empty

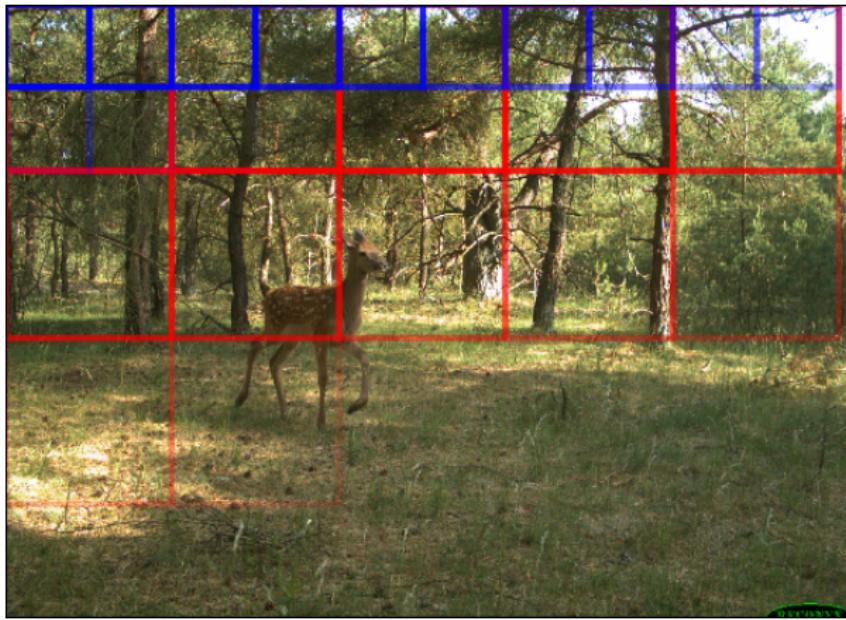


	precision	recall
--	-----------	--------

deer	0.92	0.81
badger	0.73	1.00
empty	0.67	0.60

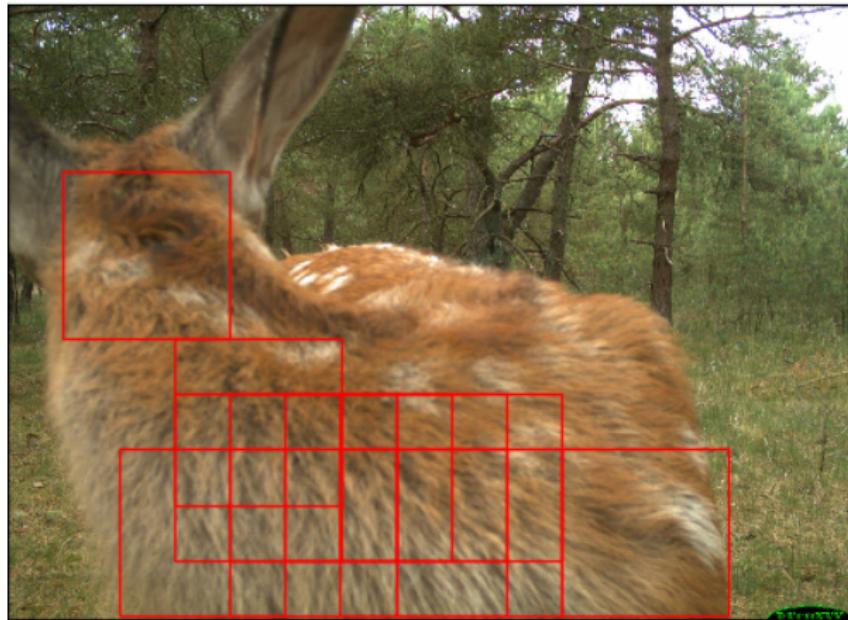
# Objektdetektion mit PCA

- ▶ Sliding Window



## Objektdetektion mit PCA

similar to 'deer'



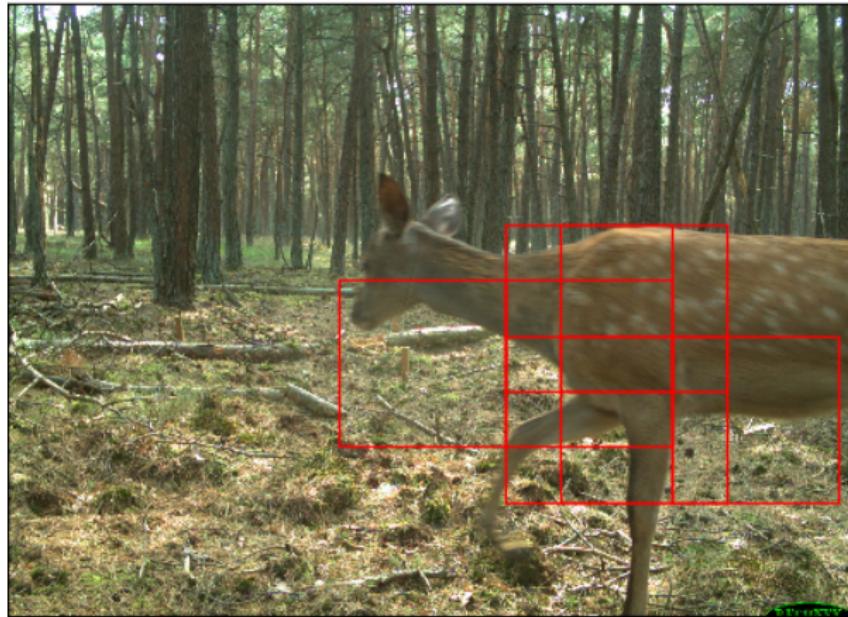
## Objektdetektion mit PCA

similar to 'deer'



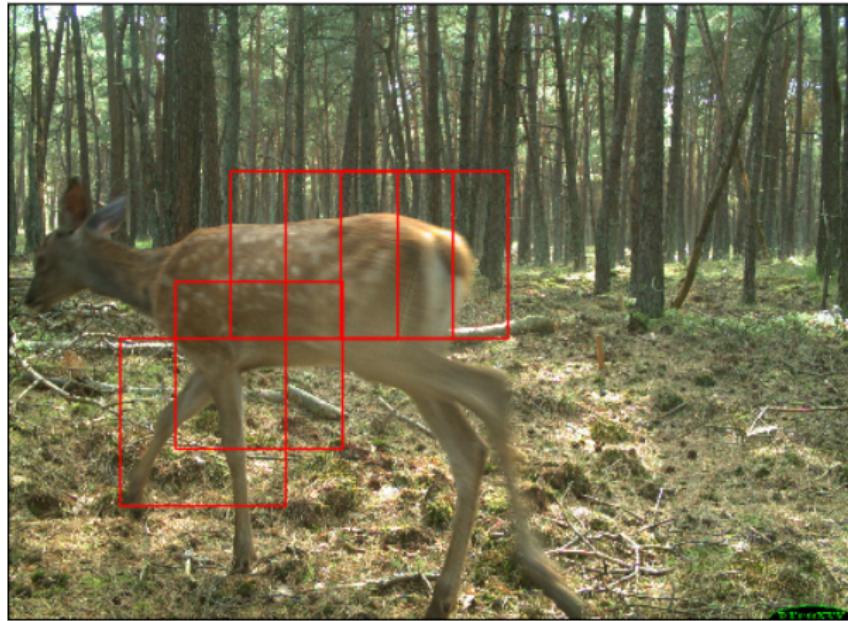
## Objektdetektion mit PCA

similar to 'deer'



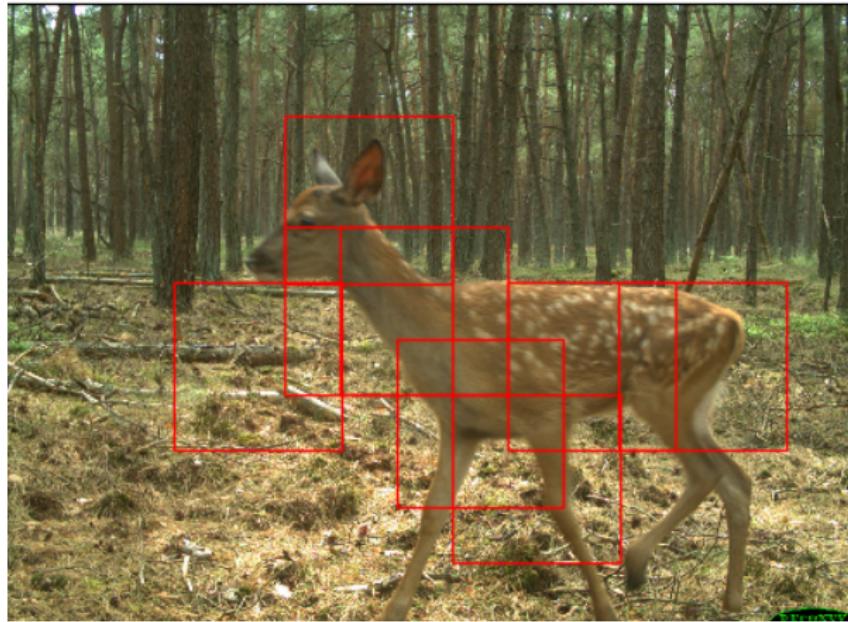
## Objektdetektion mit PCA

similar to 'deer'



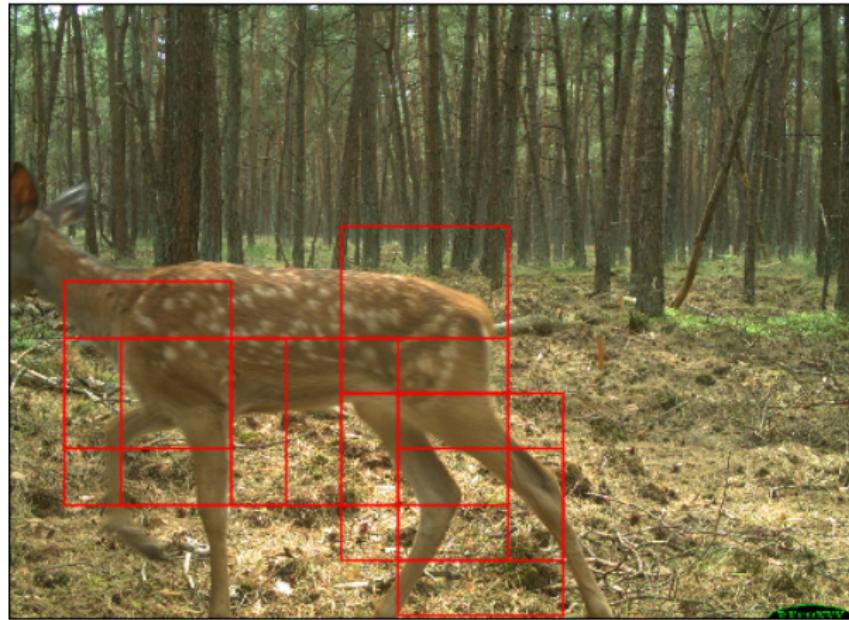
## Objektdetektion mit PCA

similar to 'deer'



## Objektdetektion mit PCA

similar to 'deer'



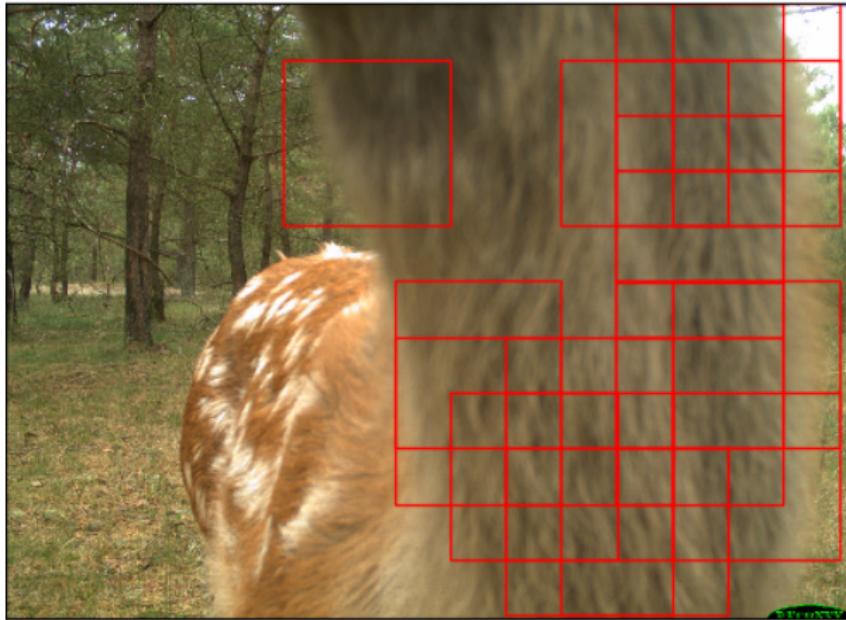
## Objektdetektion mit PCA

similar to 'deer'



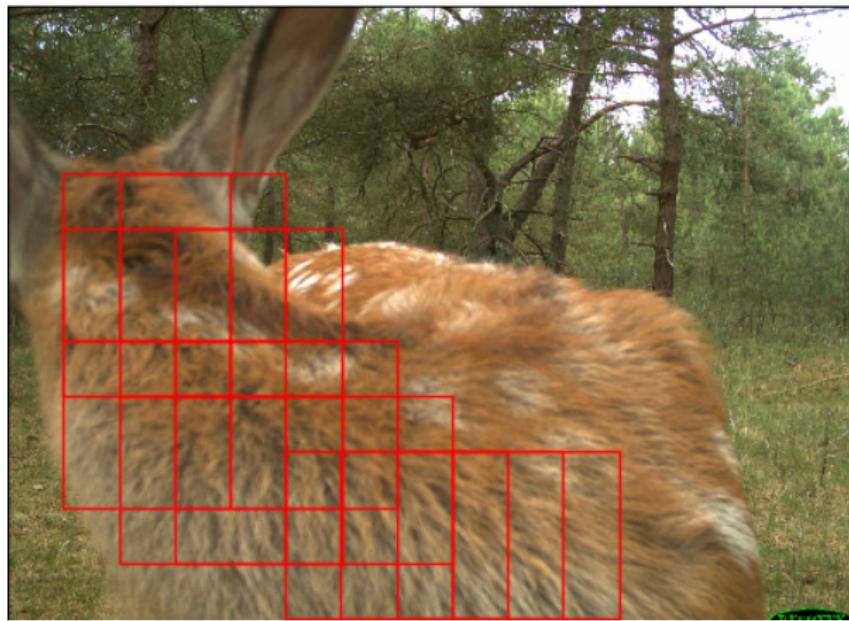
## Objektdetektion mit PCA

similar to 'deer'



## Objektdetektion mit PCA

similar to 'deer'



1. Aufteilung auf Sequenzen

2. Segmentierung und Lokalisierung

3. Lokalisierung und Klassifizierung mit HOGs und SVMs

4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs

5. Evaluierung

6. Ausblick

???

bla

1. Aufteilung auf Sequenzen
2. Segmentierung und Lokalisierung
3. Lokalisierung und Klassifizierung mit HOGs und SVMs
4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs
5. Evaluierung
6. Ausblick

???

bla

1. Aufteilung auf Sequenzen
2. Segmentierung und Lokalisierung
3. Lokalisierung und Klassifizierung mit HOGs und SVMs
4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs
5. Evaluierung
6. Ausblick

???

bla

1. Aufteilung auf Sequenzen
2. Segmentierung und Lokalisierung
3. Lokalisierung und Klassifizierung mit HOGs und SVMs
4. Klassifizierung mit Locality-constrained Linear Coding, Spatial Pyramid Matching und linearen SVMs
5. Evaluierung
6. Ausblick

## Ausblick

- ▶ PCA funktioniert nur auf Sequenzen von Bildern
  - ▶ Idee: baue während PCA Index von Bildern aller Kameras (nach Seriennummer) auf
  - ▶ benutze diesen Index zur Segmentierung von Einzelbildern, indem diese zur Sequenz vervollständigt werden
- ▶ Spatial Pyramid Matching ist momentan sehr langsam
  - ▶ Bottleneck: die Berechnung der LLC-Codes, insbesondere die Lösung des linearen Gleichungssystems
  - ▶ optimiere Code mit *numba* und parallelisiere weiter [Numba]
- ▶ Kombiniere SPM mit LLC durch verschiedene Arten von Features, wie von [Yu et al. 13] vorgeschlagen
  - ▶ berechne Textur- oder Farbfeatures, wie beispielsweise CLBP auf dichtem Gitter
  - ▶ Kombination mit SIFT-Features über *Boosting* oder Scikit-learns *Voting Classifier*

# Quellen I

- [PH03] *ExifTool by Phil Harvey.* <https://www.sno.phy.queensu.ca/~phil/exiftool/>. Accessed: 04.03.2019.
- [LSP06] Svetlana Lazebnik, Cordelia Schmid und Jean Ponce. „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.
- [Numba] *Numba.* <http://numba.pydata.org/>. Accessed: 04.03.2019.
- [Wang et al. 10] Jinjun Wang u. a. „Locality-constrained Linear Coding for Image Classification“. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010.
- [YGH09] Jianchao Yang 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 u. a. „Automated identification of animal species in camera trap images“. In: *EURASIP Journal on Image and Video Processing* (2013).