HAW HAMBURG

**BACHELORTHESIS**

Tobias Ranfft
Matrikelnummer 2202954

# Identifikation sozialer Gruppen in Videos, Motion-Similarity-Clustering und Multi-Object-Tracking unter Anwendung eines CNNs zur Identifikation von Personen in Videostreams

**FAKULTÄT TECHNIK UND INFORMATIK**
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

**HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG**
**Hamburg University of Applied Sciences**

Tobias Ranfft

# Identifikation Sozialer Gruppen von Fußgängern in Videos, Motion-Similarity-Clustering und Multi-Object-Tracking unter Anwendung eines CNNs zur Identifikation von Personen in Videostreams

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Clemen
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 14. Juli 2023

**Tobias Ranfft**

## Thema der Arbeit

Identifikation Sozialer Gruppen von Fußgängern in Videos, Motion-Similarity-Clustering und Multi-Object-Tracking unter Anwendung eines CNNs zur Identifikation von Personen in Videostreams

## Stichworte

YOLO, Object-Detection, Motion-Similarity-Clustering, Convolutional Neural Network, Bildverarbeitung

## Kurzzusammenfassung

Die Echtzeiterkennung von Personengruppen in Videos ist für diverse Anwendungs- und Forschungsbereiche zu einem relevanten Thema geworden. Hier zu nennen sind unter anderem Videoüberwachung, Verhaltensforschung und Prozessoptimierung. Ziel dieser Arbeit ist daher die Entwicklung eines Systems zur automatisierten Erkennung und Markierung von Personen und deren Gruppenzugehörigkeiten. Beschrieben wird zunächst YOLOv3 als genutztes Convolutional Neural Network. Unter heranziehen der Konzepte des NMSC-Algorithmus wird ein System zur Erkennung von Personengruppen anhand ihrer Bewegungsparameter entwickelt und beschrieben. Die Auswertung und Parametrisierung des Systems findet exemplarisch anhand eines Beispiels statt. Es wird gezeigt, dass das System Personen und Gruppen im Vergleich zur manuellen Identifikation meist zuverlässig erkennt. Ausnahmen resultieren vor allem aus der zugrundeliegenden Gruppendefinition und der gewählten Methoden. Zur Verbesserung des Systems wäre eine Modifikation in Bezug auf die zugrundeliegenden Methoden für die Erkennung und das Tracking von Personen denkbar.

**Tobias Ranfft**

**Title of Thesis**

Identification of social groups from Pedestrians in Videos, Motion-Similarity-Clustering and Multi-Object-Tracking with the usage of a CNN to identify persons in videostreams

**Keywords**

Motion-Similarity-Clustering, YOLO, Object-Detection, Convolutional Neural Network, Computer Vision

## Abstract

Real-time detection of grouped persons in videos has recently become a relevant topic in research and for various applications, e.g. video-surveillance, social studies or process-optimization. The aim of this thesis is therefore the development of a system to identify and label pedestrians and groups. Firstly, the thesis provides insights regarding the YOLOv3 architecture since its used as a detection-framework in the developed system. By using basic concepts from the NMSC-Algorithm a system to identify and follow up groups of persons through motion-similarity is developed. Its evaluation and parameterization is performed based on a demo-video. It is shown that in comparison to manual identification the system identifies persons and groups almost as accurately. Discrepancies can be explained through the applied grouping definition and through the selected methods for object-detection and -tracking. Improvements in accuracy might be achievable by modifying these.

# Inhaltsverzeichnis

**Abbildungsverzeichnis** ..... **vii**

**Abkürzungsverzeichnis** ..... **ix**

**1 Einleitung** ..... **10**

1.1 Thema ..... 10

1.2 Aufgabenstellung ..... 11

1.3 Zielgruppe ..... 11

**2 Methodik** ..... **12**

2.1 Multi-Object-Detection mit YOLO ..... 12

2.1.1 YOLO – You Only Look Once in der ersten Version ..... 12

2.1.2 Weiterentwicklung und Änderungen bis YOLOv3 ..... 17

2.1.3 Ergebnisse von YOLOv3 im Vergleich zu anderen CNNs ..... 18

2.2 Identifikation sozialer Gruppen – Motion-Similarity-Clustering ..... 20

2.2.1 Grundlagen des NMSC-Algorithmus ..... 20

2.2.2 Bewertung des NMSC-Algorithmus ..... 21

**3 Design und Implementierung** ..... **22**

3.1 Systemkontext und -anforderungen ..... 22

3.2 Systemkomponenten ..... 24

3.2.1 Detector ..... 27

3.2.2 Tracker ..... 27

3.2.3 Grouping ..... 29

**4 Auswertung** ..... **31**

4.1 Vergleich der Ergebnisse mit Erwartungswerten ..... 31

4.2 Szenenspezifische Auswertung der Gruppen ..... 36

4.2.1 Hinzufügen von Personen zu bestehenden Gruppen ..... 36

4.2.2 Auflösen von Gruppen ..... 39
4.3 Konsistenz der (Gruppen-)Identifikationsnummern ..... 40
4.4 Abgleich der Ergebnisse mit den definierten Anforderungen ..... 42
**5 Diskussion ..... 43**
5.1 Fazit ..... 43
5.2 Einschränkungen des entwickelten Systems ..... 45
5.3 Ausblick ..... 46
**Literaturverzeichnis ..... 48**
**A Anhang ..... 51**
A.1 Anhang 1: Beispielvideo ..... 51
A.2 Anhang 2: System ..... 51
A.3 Anhang 3: Ergebnisvideo ..... 51

# Abbildungsverzeichnis

Abbildung 1 - YOLOv3 Architektur (Terven/Cordova-Esparza, 2023) ..... 18
Abbildung 2 – Precision YOLOv3 im Vergleich zu anderen Multi-Object-Detection Architekturen (Redmon/Farhadi, 2018) ..... 18
Abbildung 3 - Vergleich inference-time (Redmon/Farhadi, 2018) ..... 19
Abbildung 4 - Systemkontextdiagramm ..... 23
Abbildung 5 Komponentendiagramm ..... 26
Abbildung 6 - Ausgabe „Detector“ in Frame 1 ..... 27
Abbildung 7 - Ausgabe der markierten Personen mit ID in Frame 1 ..... 28
Abbildung 8 - Ausgabe der markierten Personen mit ID in Frame 14 ..... 29
Abbildung 9 - Gruppen in Frame 40 ..... 30
Abbildung 10 – Manuelle Identifikation von Gruppen nach einer Sekunde ..... 32
Abbildung 11 – Vom System ermittelte Gruppierungen nach einer Sekunde ..... 33
Abbildung 12 - Manuelle Identifikation von Gruppen nach 3 Sekunden ..... 34
Abbildung 13 - Vom System ermittelte Gruppierungen nach 3 Sekunden ..... 34
Abbildung 14 - Manuelle Identifikation von Gruppen nach 5 Sekunden ..... 35
Abbildung 15 - Vom System ermittelte Gruppierungen nach 5 Sekunden ..... 36
Abbildung 16 - Frame 200 ..... 38
Abbildung 17 - Ausschnitt aus Frame 200 ..... 38
Abbildung 18 - Ausschnitt aus Frame 210 ..... 38
Abbildung 19 - Ausschnitt aus Frame 220 ..... 38

Abbildung 20 - Gruppen aus Frame 210 ..... 39
Abbildung 21 - Gruppen aus Frame 220 ..... 40
Abbildung 22 - Gruppen aus Frame 160 ..... 41
Abbildung 23 - Markierte Beispielgruppe ..... 42

# Abkürzungsverzeichnis

| | |
|---|---|
| **Abb.** | Abbildung |
| **AP** | Average Precision |
| **CNN** | Convolutional Neural Network |
| **grp-ID** | Gruppenidentifikationsnummer |
| **ID** | Identifikationsnummer |
| **LCSS** | Longest-Common-Subsequence-Tool |

# 1 Einleitung

## 1.1 Thema

Die automatisierte Erkennung von Personen und ihren Gruppenbildungen zeigt thematische Relevanz in diversen Bereichen: Unter anderem im Bereich der Überwachung und Sicherheit im öffentlichen Raum, sowie in der Sozialforschung findet diese ihre Anwendung (vgl. Bhargava/Chaudhuri, 2014). So kann die Erkennung von Gruppen in Menschenmassen beispielsweise zur frühzeitigen Evaluierung eventueller Sicherheitsrisiken genutzt werden (vgl. Maheshwari/Heda, 2016). Weitere mögliche Anwendungsszenarien sind außerdem die Analyse von Warteschlangen oder die Analyse gruppenbasierten Einkaufsverhaltens (vgl. Chandran et al., 2015).

Die Identifikation von Gruppen in Videos kann auf unterschiedliche Weisen erfolgen: Denkbar sind hierbei Kriterien wie unter anderem Erscheinungsbild, Bewegungsparameter oder Interaktionen zwischen Personen (vgl. Cheng et al. 2011).

Diese Arbeit befasst sich insbesondere mit der Erkennung von sozialen Gruppen in Videos anhand ihrer Bewegungsparameter als gängige Methode (vgl. ebd.). Hierbei wird ein Gruppierungs-Algorithmus entwickelt und anhand eines Beispielvideos ausgewertet. Die Erkennung der Personen im Video findet dabei mithilfe eines Convolutional Neural Network (CNN) statt. Im speziellen stellt sich heraus, das sich für das genutzte Beispielvideo YOLOv3 (vgl. Redmon/Farhadi, 2018) eignet. Um den entwickelten Algorithmus entsprechend auswerten zu können, wird eine simple Multi-Object-Tracking-Methode entwickelt und angewandt.

## 1.2 Aufgabenstellung

Ziel der Arbeit ist es ein System zur Identifikation sozialer Gruppen in Videos zu erstellen und zu bewerten.

Die Aufgabenstellung dieser Arbeit umfasst daher mehrere Teilbereiche:

1. Die Auswahl einer geeigneten Methode für die Multi-Object-Detection.
2. Die Entwicklung eines Gruppierungsalgorithmus auf Basis bestehender Konzepte.
3. Die Entwicklung eines für den Gruppierungsalgorithmus nötigen Multi-Object-Tracking-Algorithmus.
4. Die Evaluation des Systems anhand eines Beispiels.

Ziel ist es, Gruppen erfolgreich durch das System erkennen und markieren zu lassen. Personen, die keiner Gruppe angehören, sollen dabei gesondert markiert werden. Gruppen und Personen sollen über die Dauer des Beispielvideos (Anhang 1) hinweg verfolgt werden.

## 1.3 Zielgruppe

Diese Arbeit orientiert sich an Leser mit einem abgeschlossenen Informatikstudium, oder ähnlichen Vorkenntnissen, bezogen auf die Funktionsweise und Evaluationsmetriken von CNNs. Vorausgesetzt werden so unter anderem Kenntnisse in den Bereichen Bildverarbeitung, Software-Engineering und weiterführender Mathematik.

# 2 Methodik

## 2.1 Multi-Object-Detection mit YOLO

Im Bereich Multi-Object-Detection durch CNNs stehen bereits diverse Architekturen zur Verfügung (vgl. Shetty et al., 2021). Hier können generell zwei große Architektur Kategorien unterschieden werden: Einerseits die sog. *one-stage*-Algorithmen, bei welchen Input-Bilder einmalig durch ein CNN propagiert werden und dieses anschließend direkt die zugehörigen Positionen und Dimensionen als Erkennung zurückgibt (vgl. ebd.). Dem gegenüber stehen die sog. *two-stage*-Architekturen. Der elementare Unterschied zu Ersteren besteht darin, dass ein CNN zunächst nur Vorschläge für bestimmte Bereiche des Bildes ermittelt. In einem zweiten Schritt findet dann die Klassifizierung statt (vgl. ebd.).

Die hier verwendete YOLOv3-Architektur gehört zu den *one-stage*-Architekturen (vgl. Redmon et al., 2016). Im Folgenden wird YOLO als Architektur-Modell und seine Weiterentwicklung bis zur dritten Version näher erläutert.

### 2.1.1 YOLO – You Only Look Once in der ersten Version

YOLO ist eine CNN-Architektur aus dem Bereich Multi-Object-Detection, welches die Objekterkennung als Regressionsproblem betrachtet. Die ursprüngliche und daher im Folgenden näher erläuterte Version wurde von Redmon et al. (2016) entwickelt und beschrieben.

Das Input-Bild wird in YOLO einmalig durch das CNN propagiert, woraufhin direkt die zugehörigen Koordinaten der *Bounding-Boxes* ausgegeben werden und pro jeweiliger *Bounding-Box* ein *Confidence*-Wert erzeugt wird. In dieser ersten Version von YOLO – im Folgenden alternativ als YOLOv1 bezeichnet - wird das Input-Bild in ein S x S Raster eingeteilt, wobei hier jede Zelle für die Objekterkennung in dieser Zelle verantwortlich ist (vgl. ebd.).

Für jede *Bounding-Box* ermittelt das CNN die Koordinaten (x, y), welche relativ zur zugehörigen Zelle angegeben sind (vgl. Redmon et al., 2016). Des Weiteren werden die Breite w und die Höhe h in Relation zur Größe des Input-Bildes angegeben. Zusätzlich wird für jede *Bounding-Box* der *Confidence*-Wert als Pr(Object) x IoU(truth, predict) betrachtet. Pr(Object)

ist dabei diejenige Wahrscheinlichkeit, mit welcher ein Objekt in einer Zelle vorhanden ist. IoU(truth, predict) ist die Schnittfläche zwischen der vorhergesagten *Bounding-Box* und dem *ground-truth* des Datensatzes. Der *Confidence*-Wert soll daher 0 entsprechen, wenn kein Objekt erkannt wird oder, falls ein Objekt erkannt wird, die entsprechende anteilige Fläche der *Prediction* von dem *ground-truth* (vgl. Redmon et al., 2016).

Zusammengefasst werden in YOLOv1 also pro Zelle einmal die zugehörigen Klassenwahrscheinlichkeiten für die zu ermittelnden Objekte vorhergesagt. Um klassenspezifische *Confidence*-Werte zu erzeugen, werden die ermittelten *Confidence*-Werte mit den ermittelten Klassenwahrscheinlichkeiten multipliziert. Dieser Wert gibt an, mit welcher Wahrscheinlichkeit ein Objekt einer bestimmten Klasse in einer *Bounding-Box* liegt und wie genau die *Bounding-Box*-Koordinaten und -Dimensionen vorhergesagt werden (vgl. ebd.).

Daraus ergibt sich anschließend einen Vorhersagetensor von S x S x (B x 5 + C) Werten. Dabei entspricht B der Anzahl der vorhergesagten *Bounding-Boxes* pro Zelle und C der Anzahl der verschiedenen zu erkennenden Klassen (vgl. ebd.).

**Training**

Für das Training des CNNs wurde von Redmon et al. (2016) zunächst ein Teil des Netzes als Bildklassifikator auf dem *ImageNet 1000-class competition* Datensatz (vgl. Russakovsky et al., 2015) mit einer kleineren Input-Auflösung von 224 x 224 vortrainiert. Der Idee folgend, dass das Hinzufügen von *convolutional-layer* sowie *fully-connected-layer* (vgl. Lin et al., 2013) zu einem bereits vortrainierten CNN die Performanz erhöhen könne, wurde das vortrainierte CNN daraufhin um die jeweiligen Schichten erweitert und zu dem *Detection-Model* konvertiert (vgl. Redmon et al., 2016).

Die Initialisierung der Gewichte in den hinzugefügten Schichten erfolgte randomisiert (vgl. ebd.). Da der Erkennungsvorgang feinkörnige visuelle Informationen benötigt, wurde die Input-Auflösung auf 448 x 448 erhöht. Das so entstandene Modell wurde anschließend auf den öffentlich zugänglichen Datensätzen *PASCAL VOC 2007* (Everingham et al., 2007) und *PASCAL VOC 2012* (Everingham et al., 2012) als Objekterkennungsmodell trainiert (vgl. Redmon et al., 2016). Hierbei nutzten die Autor*Innen eine *Batch-size* von 64, ein *Momentum* von 0.9 sowie ein *decay* von 0.0005. Es wurde mit einer Lernrate von $10^{-3}$ begonnen, diese wurde über die ersten Epochen auf $10^{-2}$ gesteigert. Dadurch konnte ein zu frühes Divergieren

durch instabile Gradienten verhindert werden. Die Lernrate wurde nach 75 Epochen wieder auf $10^{-3}$ verringert. Nach weiteren 30 Epochen wurde die Lernrate auf $10^{-4}$ verringert. Diese blieb dann für die letzten 30 Epochen bestehen (vgl. Redmon et al., 2016). Um ein *overfitting* zu verhindern, nutzten die Autor*Innen einen *dropout-layer* mit einer Rate von 0.5 sowie Data-Augmentation (vgl. ebd.).
Ziel des Trainings war, dass die *Bounding-Boxes* korrekt lokalisiert und klassifiziert werden. Zusätzlich sollte dabei die vorhergesagte *Confidence* für ein positives Ergebnis möglichst hoch sein. Daraus ergab sich für die Autor*Innen eine *Loss*-Funktion, die aus der Summe aus den im Folgenden näher beschriebenen drei *Loss*-Funktionen besteht: Dem *Localisation Loss*, dem *Classification Loss* und dem *Confidence Loss* (vgl. ebd.).

**Localisation Loss**

Der *Localisation Loss* (Formel 1) wird über die Abweichung der Dimensionen w und h und der Positionskoordinaten x und y vom zugehörigen *ground-truth* ($\hat{x}, \hat{y}, \hat{w}, \hat{h}$) wie folgt bestimmt (vgl. Redmon et al., 2016):

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (1)$$

dabei gilt:

$\mathbb{1}_{ij}^{obj}$ = 1, *Bounding-Box* j in Zelle i für die Erkennung des Objektes verantwortlich ist

$\mathbb{1}_{ij}^{obj}$ = 0, *Bounding-Box* j in Zelle i nicht für die Erkennung des Objektes verantwortlich ist

$\lambda_{coord} = 5$

Da in Formel 1 nur diejenigen Boxes berücksichtigt wird, welche auch für die Vorhersage verantwortlich ist, werden diese über den Faktor $\lambda_{coord}$ höher Gewichtet, um eine Balance zwischen den *Losses* herzustellen. Die Abweichung über die Dimensionen w und h wird über die Wurzel der Differenzen berechnet, um die Ungleichheit zwischen großen und kleinen *Bounding-Boxes* in der Bewertung zu verringern (vgl. Redmon et al., 2016).

**Classification Loss**

Der *Classification Loss* (Formel 2) wird über die Abweichung der ermittelten Klasse für eine Box vom *ground-truth* bestimmt (ebd.).

$$\sum_{\mathbf{i=0}}^{\boldsymbol{S^2}} \mathbb{1}_i^{\boldsymbol{obj}} \sum_{\boldsymbol{c \epsilon classes}} (\boldsymbol{p_i(c)} - \boldsymbol{\hat{p}_i(c)})^{\mathbf{2}} \tag{2}$$

dabei gilt:

$\mathbb{1}_i^{obj}$ = 1, wenn ein Objekt in Zelle i vorhanden ist

$\mathbb{1}_i^{obj}$ = 0, wenn kein Objekt in Zelle i vorhanden ist

$\hat{p}_i(c)$ = klassenspezifische *Confidence* der Klasse c in Zelle i

Die Abweichungen der Klassenspezifischen *Confidence*-Werte für alle Klassen findet also nur im *Loss* Berücksichtigung, wenn ein Objekt in der entsprechenden Zelle vorhanden ist.

**Confidence Loss**

Der *Confidence Loss* (Formel 3) beschreibt die Abweichung der ermittelten Wahrscheinlichkeit, dass ein Objekt in einer Zelle vorhanden ist, vom *ground-truth* (vgl. Redmon et al., 2016):

$$\sum_{\boldsymbol{i=0}}^{\boldsymbol{S^2}} \sum_{\boldsymbol{j=0}}^{\boldsymbol{B}} \mathbb{1}_{\boldsymbol{ij}}^{\boldsymbol{obj}} (\boldsymbol{C_i} - \boldsymbol{\hat{c}_i})^{\mathbf{2}} + \boldsymbol{\lambda_{noob\,j}} \sum_{\boldsymbol{i=0}}^{\boldsymbol{S^2}} \sum_{\boldsymbol{j=0}}^{\boldsymbol{B}} \mathbb{1}_{\boldsymbol{ij}}^{\boldsymbol{noobj}} (\boldsymbol{C_i} - \boldsymbol{\hat{c}_i})^{\mathbf{2}} \tag{3}$$

dabei gilt:

$\mathbb{1}_{i\,j}^{obj}$ = 1, wenn die Box j in Zelle i für die Erkennung des Objektes verantwortlich ist

$\mathbb{1}_{i\,j}^{obj}$ = 0, wenn die Box j in Zelle i nicht für die Erkennung des Objektes verantwortlich ist

$\mathbb{1}_{i\,j}^{noobj}$ = 1, wenn die Box j in Zelle i nicht für die Erkennung des Objektes verantwortlich ist

$\mathbb{1}_{i\,j}^{noobj}$ = 0, wenn die Box j in Zelle i für die Erkennung des Objektes verantwortlich ist

$\lambda_{noobj} = 0.5$

$\hat{C}_i$ = *Confidence*-Wert der Box j in Zelle i, dass ein Objekt in der Zelle vorhanden ist

So werden von den Autor*Innen folgende Fälle unterschieden:

1. Eine Box ist für die jeweilige Objekterkennung verantwortlich: Trifft dies zu, geht der *Loss* voll in die Berechnung ein.
2. Eine Box ist nicht für die jeweilige Objekterkennung verantwortlich: Der *Loss* geht nur zu 50% in die Bewertung ein.

**Grenzen der ersten Version**

Zusammenfassend stellt YOLOv1 einen *one-stage*-Architekturen dar, welcher hinsichtlich der Benutzung unter Echtzeitvoraussetzungen geeignet ist (vgl. Shetty et al., 2021).

Ein nicht zu vernachlässigendes Problem in der initialen Version von YOLO besteht jedoch darin, dass kleine Objekte, welche nah beieinander sind, nur schwer erkannt werden (vgl. Terven/Cordova-Esparza, 2023). Das liegt daran, dass eine Zelle immer nur eine begrenzte Anzahl von Objekten erkennen kann. Zusätzlich generalisiert das Netz nur unzureichend, wenn die zu erkennenden Objekte sich in ihren Dimensionen zu stark von den Objekten aus dem Trainingsdatensatz unterscheiden (vgl. ebd.). Zuletzt zeigt sich auch das Problem der Ungleichheit von großen und kleinen Objekten: Während kleine Fehler in großen Objekten auch nur zu einem kleinen Fehler in der *Loss*-Funktion führen, enden kleine Fehler in kleinen Objekten in einem großen Fehler bei der Schnittflächen-Berechnung dieses Objektes, was dann zu stark in der *Loss*-Funktion berücksichtigt wird (vgl. Redmon et al., 2016).

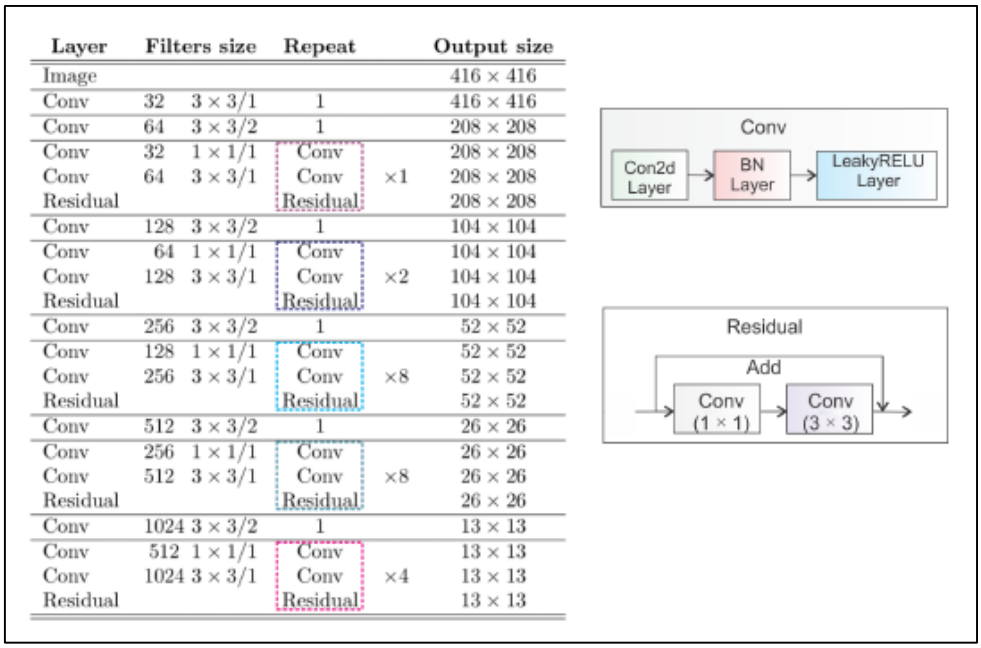
### 2.1.2 Weiterentwicklung und Änderungen bis YOLOv3

Mit YOLOv2 stellten Redmon und Farhadi (2016) noch im selben Jahr eine weiterentwickelte Version vor: Fehler in der Lokalisation konnten vermindert werden und es erfolgte eine Anpassung der Schichten und des Trainings (vgl. Redmon/Farhadi, 2016).

So wurden unter anderem sog. *Anchor-Boxes* (vgl. Ren et al., 2016) hinzugefügt: Hierdurch konnten die Dimensionen von zu erkennenden Objekten bereits im Vorhinein festgelegt werden (vgl. Redmon/Farhadi, 2016). Durch *Batch-Normalization* (vgl. Ioffe/Szegedy, 2015) konnte zudem eine verbesserte Generalisierung, eine verbesserte Trainingsgeschwindigkeit und eine automatische Regulierung erreicht werden (vgl. Redmon/Farhadi, 2016).

In YOLOv3 wurden schließlich Änderungen an der Netzwerkarchitektur vorgenommen, die *Loss*-Funktion angepasst und *multi-label-classification* hinzugefügt (vgl. Redmon/Farhadi, 2018). Ein Fokus wurde zudem auf die Tiefe des CNNs gelegt: Hier zeigte sich jedoch, dass das reine Hinzufügen von Schichten ab einer bestimmten Menge nicht weiter dazu führt, dass die Fehlerrate im Training sinkt, sondern stattdessen stagniert oder sogar wieder steigt (vgl. He et al., 2015). Gelöst wurde diese Problematik durch das Hinzufügen von sog. *residual-connections*. Durch diese wird es einem CNN möglich ein *identity-mapping* über Schichten hinweg anzunähern (vgl. ebd.).

Die untenstehende Abbildung (Abb. 1) zeigt die Netzwerkarchitektur von YOLO in der dritten Version. Hier auch zu sehen ist die Anwendung der *Batch-Normalization* und den *residual-connections*.



| Layer | Filters | size | Repeat | | Output size |
|---|---|---|---|---|---|
| Image | | | | | 416 × 416 |
| Conv | 32 | 3 × 3/1 | 1 | | 416 × 416 |
| Conv | 64 | 3 × 3/2 | 1 | | 208 × 208 |
| Conv | 32 | 1 × 1/1 | Conv | | 208 × 208 |
| Conv | 64 | 3 × 3/1 | Conv | ×1 | 208 × 208 |
| Residual | | | Residual | | 208 × 208 |
| Conv | 128 | 3 × 3/2 | 1 | | 104 × 104 |
| Conv | 64 | 1 × 1/1 | Conv | | 104 × 104 |
| Conv | 128 | 3 × 3/1 | Conv | ×2 | 104 × 104 |
| Residual | | | Residual | | 104 × 104 |
| Conv | 256 | 3 × 3/2 | 1 | | 52 × 52 |
| Conv | 128 | 1 × 1/1 | Conv | | 52 × 52 |
| Conv | 256 | 3 × 3/1 | Conv | ×8 | 52 × 52 |
| Residual | | | Residual | | 52 × 52 |
| Conv | 512 | 3 × 3/2 | 1 | | 26 × 26 |
| Conv | 256 | 1 × 1/1 | Conv | | 26 × 26 |
| Conv | 512 | 3 × 3/1 | Conv | ×8 | 26 × 26 |
| Residual | | | Residual | | 26 × 26 |
| Conv | 1024 | 3 × 3/2 | 1 | | 13 × 13 |
| Conv | 512 | 1 × 1/1 | Conv | | 13 × 13 |
| Conv | 1024 | 3 × 3/1 | Conv | ×4 | 13 × 13 |
| Residual | | | Residual | | 13 × 13 |



Abbildung 1 - YOLOv3 Architektur (Terven/Cordova-Esparza, 2023)

### 2.1.3 Ergebnisse von YOLOv3 im Vergleich zu anderen CNNs

In Abbildung 2 und Abbildung 3 sind Performanz und Genauigkeit der von Redmon und Farhadi (2018) verglichenen Multi-Object-Detection-Architekturen abgebildet. Dabei ist zu erkennen, dass YOLOv3 ein performantes Netzwerk ist ohne große Schwächen in der Präzision aufzuzeigen (vgl. ebd.).

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

Abbildung 2 – Precision YOLOv3 im Vergleich zu anderen Multi-Object-Detection Architekturen (Redmon/Farhadi, 2018)

Die Präzision eines CNN wird anhand verschiedener Metriken ermittelt: Hier wird die Average Precision (AP) unter anderem für kleine (APs), mittelgroße ($AP_M$) und große Objekte ($AP_L$) separat bestimmt und kann so für unterschiedliche Anwendungsszenarien zum Vergleich verschiedener Multi-Object-Detection Architekturen herangezogen werden (vgl. Koech, 2020).

Abbildung 2 zeigt, dass YOLOv3 im Vergleich zur Vorgängerversion YOLOv2 die angestrebten Verbesserungen in der Erkennung kleiner Objekte erreicht.

Im Vergleich zu allen anderen *one-stage*-Architekturen zeigt YOLOv3 vor allem in der Erkennung von großen Objekten schlechtere Ergebnisse als der Rest. Hier ist der $AP_L$ von YOLOv3 bis zu 10,2 schlechter als der, der anderen. In der Erkennung von mittelgroßen Objekten liegt YOLOv3 ungefähr im selben Bereich wie SSD513 (vgl. Liu et al. 2016) und DSSD513 (vgl. Fu et al., 2017).

| Method | mAP | time |
|---|---|---|
| [B] SSD321 | 28.0 | 61 |
| [C] DSSD321 | 28.0 | 85 |
| [D] R-FCN | 29.9 | 85 |
| [E] SSD513 | 31.2 | 125 |
| [F] DSSD513 | 33.2 | 156 |
| [G] FPN FRCN | 36.2 | 172 |
| RetinaNet-50-500 | 32.5 | 73 |
| RetinaNet-101-500 | 34.4 | 90 |
| RetinaNet-101-800 | **37.8** | 198 |
| **YOLOv3-320** | 28.2 | **22** |
| **YOLOv3-416** | 31.0 | 29 |
| **YOLOv3-608** | 33.0 | 51 |

Abbildung 3 - Vergleich inference-time (Redmon/Farhadi, 2018)

Hervorzuheben ist im Vergleich vor allem die *inference-time*: Hier liegt YOLOv3 wie in Abbildung 3 erkennbar mit Werten zwischen 22ms und 51ms vorn. (Redmon et al., 2018)

Zusammenfassend bietet YOLOv3 also bei ähnlicher Genauigkeit im Bereich der mittelgroßen Objekte gleichzeitig eine kurze *inference-time*. Aus diesem Grund wird YOLOv3 in dieser Arbeit als Multi-Object-Detection-Architektur genutzt.

## 2.2 Identifikation sozialer Gruppen – Motion-Similarity-Clustering

Im Folgenden wird auf die hier als Grundlage verwendete Methode von Chandran et al. (2015) näher eingegangen. Der entwickelte Algorithmus wurde von den Autor*Innen als *Non-Recursive-Motion-Similarity-Clustering*-Algorithmus (NMSC-Algorithmus) benannt (vgl. Chandran et al., 2015).

### 2.2.1 Grundlagen des NMSC-Algorithmus

Die Kernidee hierbei ist es, Fußgänger anhand von Positions- und Bewegungsdaten unter folgenden Kriterien in Gruppen einzuteilen:

1. Abstand: $s < 2$ [m]
2. Geschwindigkeitsunterschied: $\Delta v < 0.4$ [m/s]
3. Bewegungsrichtungsunterschied: $\Delta d < 3°$

Diese Kriterien stammen aus der sozialpsychologischen Untersuchung des Verhaltens von Fußgängern in Gruppen (vgl. Chandran et al., 2015).

Die Autor*Innen berechnen zunächst Positionen und Geschwindigkeiten für alle Personen. In jedem Bild eines Videos wird für jede Person ein Vektor der Form $f_i^t = (x_i^t, y_i^t, u_i^t, v_i^t)^T$ erstellt. Hierbei sind $(x_i^t, y_i^t)$ die Positions-Koordinaten und $(u_i^t, v_i^t)$ die entsprechenden Geschwindigkeiten in x- und y-Richtung einer Person i zum Zeitpunkt t (vgl. ebd.). Die Bewegungsrichtung ist somit implizit über die Geschwindigkeiten abgebildet. Der Abstand ergibt sich dann direkt aus der Differenz der Positionen zweier Personen. Zur Ermittlung der Ähnlichkeit zwischen den Geschwindigkeiten und Richtungen zweier Personen (i, j) wird wie folgt die Differenz der Geschwindigkeiten (Formel 4) berechnet (vgl ebd.):

$$\Delta V_i = (u_i, v_i) - (u_j, v_j) \tag{4}$$

Bewegungsgeschwindigkeit und -richtung gelten dann unter folgender Bedingung (Formel 5) als ähnlich (gemäß den oben definierten Kriterien):

$$||\Delta V_i||_\infty < V_\epsilon \tag{5}$$

wobei $V_\epsilon$ ein zu ermittelnder Grenzwert ist (vgl. Chandran et al., 2015). Die Autor*Innen begründeten diesen Schritt damit, dass diese Methode weniger anfällig auf Rauschen in den ermittelten Geschwindigkeitsdaten und Bewegungsrichtungen reagiert. Um die Auswirkungen des Rauschens in den Datensätzen weiter zu vermindern wird zusätzlich das *Longest-Common-Subsequenz-tool* (LCSS-Algorithmus) genutzt (vgl. ebd.). Hierbei müssen die Daten zweier Personen nicht zu jedem Zeitpunkt als ähnlich interpretiert werden um einen Ähnlichkeitswert zu erhalten. Die ermittelten Werte aus dem LCSS-Algorithmus werden anschließend statistisch weiterverarbeitet, woraufhin dann eine Gruppierung von Personen stattfindet. Personen werden gruppiert, wenn die Ähnlichkeit über einen definierten Zeitraum erhalten bleibt (vgl. Chandran et al., 2015).

### 2.2.2 Bewertung des NMSC-Algorithmus

Zur Evaluation des Algorithmus wurde der *kappa-score* (vgl. Berry, 1992) genutzt. Dieser vergleicht die vom System identifizierten mit von Menschen identifizierten Gruppen.

Der ermittelte *kappa-score* von 0,78 zeigt auf, dass das System in der Lage ist, Gruppen zumeist wie ein Mensch zu erkennen (vgl. Chandran et al., 2015).

Als weiterer Vorteil wird von den Autor*Innen die Erkennung von Personengruppen beliebiger Größe angeführt. Die deutlich verbesserte Rechenzeit im Vergleich zu den konkurrierenden Algorithmen ist über die geringere Komplexität des Algorithmus erklärt. So hat der NMSC-Algorithmus eine Komplexität von $O(n^2)$ (vgl. ebd.). Der von Chandran et al. (2015) verglichene Konkurrenz-Algorithmus von Wirz et al. (2011), welcher ähnlich wie der NMSC-Algorithmus dazu in der Lage ist beliebig große Gruppen zu identifizieren, weist eine Komplexität von $O(n^3)$ auf (vgl. ebd.).

# 3 Design und Implementierung

Nachfolgend wird das in dieser Arbeit entwickelte System und die dazugehörigen Teilsysteme zur Identifikation sozialer Gruppen in Videos näher beschrieben.

Das System wurde auf Windows 10 22H2 in Python 3.10 entwickelt. Zusätzlich wurde *OpenCV* in der Version 4.8.0 und *NumPy* in der Version 1.25.0 zur Analyse und Videoverarbeitung genutzt.

Das System ist im Anhang referenziert und wird in der digitalen Version mit eingereicht (Anhang 2).

## 3.1 Systemkontext und -anforderungen

Der Systemkontext (Abb. 4) besteht aus einem zu analysierenden Video und den zugehörigen Metainformationen hierzu. Metainformationen beinhalten diejenigen Informationen, welche für die Transformation der Bildpunkte in ein homogenes Koordinatensystem benötigt werden.

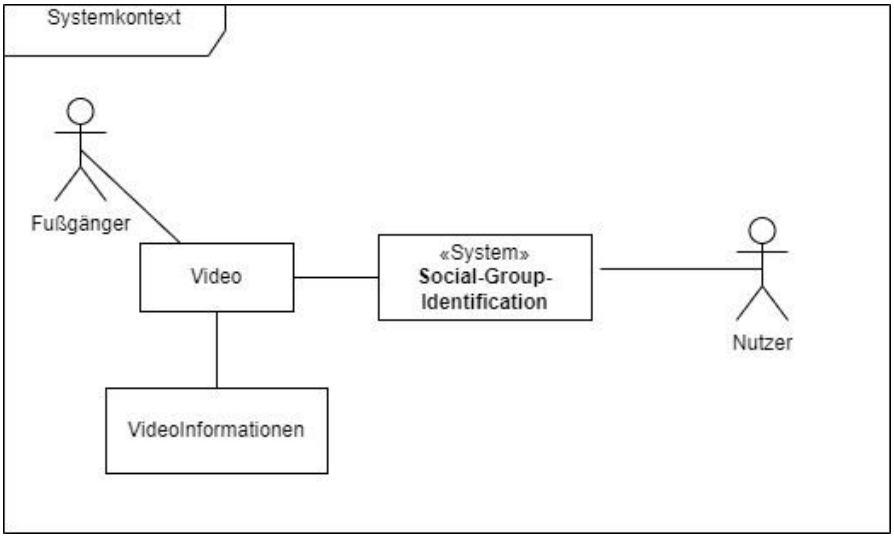



Abbildung 4 - Systemkontextdiagramm

Aus dem vorgestellten Systemkontext und der beschriebenen Aufgabenstellung resultieren folgende Anforderungen an das System:

1. Das System liest Videos ein.
2. Das System erkennt Personen in Frames und gibt erkannten Personen-Identifikationsnummern (ID).
3. Das System kann aktuell erkannte Personen den erkannten Personen aus dem vorigen Frame zuordnen.
4. Das System ordnet allen Personen, die keiner Person aus dem vorherigen Frame zugeordnet werden können, eine neue ID zu.
5. Das System errechnet aktuelle Positionen und Geschwindigkeiten für aktuell erkannte Personen.
6. Das System gruppiert Personen auf Basis der jeweiligen Positions- und Geschwindigkeitsdaten und vergibt entsprechende Gruppen-Identifikationsnummern (grp-ID).
7. Die grp-IDs bleiben über die Frames hinweg erhalten.
8. Wenn eine Gruppe sich aufteilt, erhält eine verbliebene Teilgruppe die alte grp-IDs. Der anderen Teilgruppe wird eine neue grp-ID zugeordnet.
9. Personen, die keiner Gruppe zugehören, werden als „nicht gruppiert“ erkannt und erhalten die grp-ID 0.

10. Das System markiert die bestehenden Gruppen in jedem Bild.
11. Das System generiert ein Video aus den Bildern mit den markierten Gruppen.

## 3.2 Systemkomponenten

Wesentliche Komponenten, welche für die Umsetzung der Aufgabenstellung (Abschnitt 1.2) nötig sind, sind in Abbildung 5 dargestellt.

Der Input besteht aus einem zu analysierenden Video mit den zugehörigen Videoinformationen. Die Videoinformationen bestehen aus:

1. Anzahl der aufgenommenen Bilder pro Sekunde
2. Dimensionen = Breite und Höhe der Bilder im Video

**Input**

Die *Transformationskoordinaten* bestehen aus je zwei Sätzen mit je vier Bildpunktkoordinaten. Mithilfe dieser werden die Bildpunkte des Inputbildes auf Bildpunkte des Zielbildes so projiziert, dass das Koordinatensystem des Zielbildes homogenisiert wird. Für die Transformation wurden Methoden aus der Bildverarbeitungsbibliothek *OpenCV* verwendet (Bradski, 2000).

**System**

Der „Detector“ ist als Komponente für die Erkennung von Personen in den einzelnen Frames verantwortlich. Hierfür wird das vortrainierte CNN YOLOv3 (siehe Abschnitt 2.1) genutzt, welches für die einzelnen Frames des Videos zugehörige *Bounding-Boxes* für erkannte Personen bestimmt.

Die Komponente „Tracker“ ist für die Vergabe von IDs an erkannte Personen und für die Reidentifizierung der Personen in aufeinanderfolgenden Frames verantwortlich.

Die „Grouping“-Komponente ermittelt für alle aktuell erkannten Personen zunächst die Position und Geschwindigkeit. Auf Basis dieser ermittelten Werte analysiert die Komponente daraufhin die Ähnlichkeit aller Personen im Bild. Die Komponente vergibt anschließend gleiche grp-IDs an Personen, wenn diese den Gruppierungsparametern zueinander entsprechen.

Die „Controller“-Komponente nutzt die „VideoProcessor“-Komponente, um die Bilder aus dem Inputvideo zu extrahieren. Sie verwaltet schließlich die Aktivitäten der beschriebenen aktiven Komponenten „Detector“, „Tracker“ und „Grouping“. Zusätzlich speichert sie für jedes Bild die aktuell erkannten Personen sowie über alle Bilder hinweg alle verlorenen (also nicht mehr wiedererkannten) Personen. Personen werden dann in jedem Bild anhand ihrer grp-ID markiert. Diese Markierung findet zum einen über einen Text in der Form (<ID> // <grp-ID>) und zum anderen farblich statt. Das so entstandene Bild wird dann gespeichert. Die „Controller“-Komponente nutzt am Ende der Analyse die Komponente „VideoConverter“ um aus den gespeicherten Bildern ein Outputvideo zu generieren.

Die aktiven Komponenten „Detector“, „Tracker“ und „Grouping“ werden nun, da sie für die aktive Bildverarbeitung die relevanten Teilsysteme darstellen, näher erläutert.

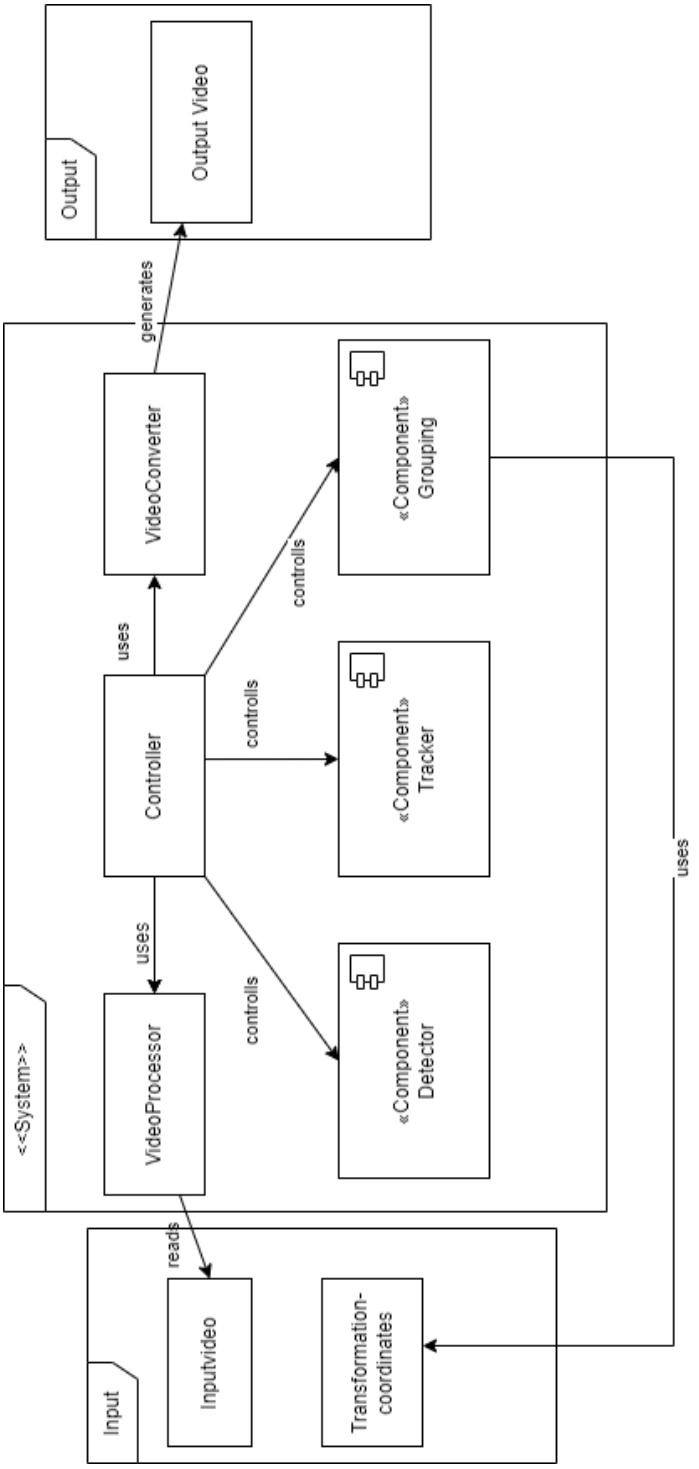



Abbildung 5 Komponentendiagramm

### 3.2.1 Detector

Aus der Ausgabe der YOLOv3-Architektur werden nur diejenigen *Detections* genutzt, welche eine Person mit einem *Confidence*-Wert von 0.5 oder höher erkennen. Zusätzlich wurde die *Non-Maxima-Supression*-Methode aus *OpenCV* (Bradski, 2000) angewandt, um gegebenenfalls doppelt erkannte Personen zu eliminieren. Die so gefilterten Personen werden dann von der „Tracker"-Komponente weiterverwertet. In Abbildung 6 sind die resultierenden *Detections* in Form von weißen Rechtecken markiert.

Abbildung 6 - Ausgabe „Detector" in Frame 1

### 3.2.2 Tracker

Der „Tracker" nutzt in jedem Frame die aktuellen *Detections* der Komponente „Detector". Der Algorithmus berechnet die Euklidische Distanz von den Zentren der aktuellen *Detections* zu allen Zentren der zuvor erkannten Personen. Ist diese Distanz minimal und gleichzeitig kleiner der experimentell für das Beispielvideo ermittelten Grenze von 10, wird die *Detection* dieser Person zugeordnet.

Für alle Personen, denen keine *Detection* zugeordnet wurde, wird über maximal vier folgende Frames eine *Prediction* vorgenommen. Hierbei wird das Zentrum um denjenigen Abstand verschoben, um welchen sich das Zentrum zuletzt verschoben hat. Die ermittelten Dimensionen zur betreffenden Person werden dabei aus der vorigen *Detection* übernommen. Personen, die über vier Bilder hinweg *predicted* sind, werden als *lost* markiert. Wenn eine aktuelle *Detection* keiner Person zugeordnet werden kann, wird die *Detection* als neue Person interpretiert. Diese Person erhält dann eine entsprechende neue ID.

Der Algorithmus gibt jeweils eine Liste aus als *lost* markierten Personen, aus *tracked* Personen und aus *predicted* Personen zurück.

In Abbildung 7 ist zu erkennen, dass der „Tracker“ jeder Person im ersten Bild des Videos eine entsprechend einzigartige ID zugewiesen hat. Dass IDs im Verlauf des Videos erhalten bleiben ist in Abbildung 8 zu sehen.


Abbildung 7 - Ausgabe der markierten Personen mit ID in Frame 1


Abbildung 8 - Ausgabe der markierten Personen mit ID in Frame 14

### 3.2.3 Grouping

Zunächst werden mithilfe der Transformationskoordinaten Positionen und Geschwindigkeiten für alle Personen berechnet. Hierbei wird das Zielbild auf Basis einer bekannten Referenzfläche auf dem Boden des Quellbilds erstellt. Aufgrund dessen wird die Position einer Person wie folgt interpretiert: Die Position einer Person entspricht der Mitte derjenigen Seite einer *Detection*, an welcher sich die Füße der Person befinden.

Die Position wird anhand der aktuellen *Detection* einer Person ermittelt. Als Geschwindigkeit wird das Mittel der Geschwindigkeiten über die letzte Sekunde genutzt. Auf Basis dieser ermittelten Positions- und Bewegungsdaten werden Ähnlichkeiten zwischen Personen berechnet. Als ähnlich werden die Bewegungs- und Positionsdaten zweier Personen nach den Kriterien aus Abschnitt 2.2.1 interpretiert. Besteht eine Ähnlichkeit zwischen zwei Personen länger als eine definierte zeitliche Grenze, werden diese Personen einer gemeinsamen Gruppe zugeordnet. Eine Ähnlichkeit erhält initial die Parameter *firstDetectedFrameNo* und *age*. Die *age*-Variable wird in jedem verarbeiteten Frame um Eins erhöht und immer dann auf 0 zurückgesetzt,

wenn die Ähnlichkeit in einem Bild gefunden wird. Die Ähnlichkeit wird entfernt, wenn *age* einen definierten Grenzwert überschreitet oder aber eine der zugehörigen Personen nicht mehr *tracked* ist.

Die so ermittelten Ähnlichkeiten werden daraufhin in jedem Frame analysiert. Aus den Ähnlichkeiten werden Gruppen anhand folgender Bedingung gebildet: Die aktuelle Frame-Nummer abzüglich der *firstDetectedFrameNo* einer Ähnlichkeit ist größer einer definierten zeitlichen Grenze.

So müssen Personen – ähnlich der Systematik des NMSC-Algorithmus (Abschnitt 2.2.1) – nicht in jedem Frame Ähnlichkeiten aufweisen, jedoch aber über einen definierten Zeitraum hinweg. Der so implementierte Algorithmus ist also weniger störanfällig auf Rauschen in den Positions- und Bewegungsdaten. Dieses Rauschen resultiert vornehmlich durch den „Detector“.

In Abbildung 9 ist exemplarisch zu sehen, wie die gruppierten Personen in Bildern markiert sind.


Abbildung 9 - Gruppen in Frame 40

# 4 Auswertung

Die Auswertung der vom System ermittelten Gruppen erfolgte anhand eines Beispielvideos (Anhang 1). Dieses beinhaltet 341 Frames und wurde mit einer Rate von 25 Bildern pro Sekunde bei einer Auflösung von 640x360 Pixeln erstellt.

Zu prüfen war, ob das System im Verlauf des Beispielvideos sinnvolle Gruppen erkennt und auch verfolgt. So sollen nach den oben beschriebenen Anforderungen aus Abschnitt 3.1 IDs und grp-ID zu Personen zugeordnet werden und erhalten bleiben. Zusätzlich wurde untersucht, ob Personen in einer erkannten Gruppe entsprechend markiert werden. Die Markierung soll sowohl über eine grp-ID als auch farblich stattfinden. Verlässt eine Person eine Gruppe, so soll diese entweder die grp-ID 0 (wenn sie fortan keiner Gruppe mehr zugehört) oder eine entsprechende grp-ID für die neue Gruppe erhalten.

## 4.1 Vergleich der Ergebnisse mit Erwartungswerten

Zur Erstellung von Erwartungswerten wurden manuell Gruppen in dem Beispielvideo identifiziert.

Hierzu wurden exemplarisch drei Zeitpunkte ausgewählt: Nach einer, drei und fünf Sekunden Videoverlauf. Für diese Zeitpunkte wurden auf den entsprechenden Frames Markierungen (gelb) eingefügt. Anschließend erfolgte ein quantitativer und qualitativer Vergleich zwischen manueller und systemgesteuerter Gruppenidentifikation.

**Vergleich der Erkennung nach einer Sekunde Videozeit**

Abbildung 10 zeigt die manuellen Gruppenidentifikationen, Abbildung 11 die durch das System markierten Gruppen zum selben Zeitpunkt. Im Vergleich der beiden Abbildungen zeigte sich, dass die vom System erkannte Gruppenanzahl vom Erwartungswert abweichte. Vier der erwarteten fünf Gruppen wurden vom System erkannt. Das System identifizierte insgesamt 7 Gruppen. Somit hat das System drei Gruppen als solche identifiziert, die nicht in der Erwartungsmenge enthalten waren.

Als weiterer Vergleichsparameter konnte die identifizierte Gruppengröße herangezogen werden. So war zu erkennen, dass das System in der Gruppe mit der grp-ID 1 nur drei von vier Personen *getrackt* hat. Dementsprechend wurden also nur die drei *getrackten* Personen der zugehörigen Gruppe zugeordnet. Ursächlich hierfür waren fehlende *Detections* aus der „Detector"-Komponente: Personen, welche vom „Detector" nicht als solche erkannt wurden, konnten auch nicht *getrackt*, und somit auch keiner Gruppe zugeordnet werden.

Der Gruppe mit der grp-ID 4, in Abbildung 11 in hellgrün zu sehen (vgl. auch Gruppe 4 in Abb. 13), wurde vom System eine Person mehr hinzugefügt als erwartet.

Abbildung 10 – Manuelle Identifikation von Gruppen nach einer Sekunde


Abbildung 11 – Vom System ermittelte Gruppierungen nach einer Sekunde

**Vergleich der Erkennung nach drei Sekunden Videozeit**

Im Vergleich der Bilder nach drei Sekunden (Abb. 12 und Abb. 13) ist zu sehen, dass auch hier die Anzahl der erwarteten Gruppen (4) von den durch das System identifizierten Gruppen (9) differiert. Dabei wurde eine der erwarteteten Gruppen nicht vom System erkannt. Zusätzlich durch das System erkannte Gruppen waren die Gruppen mit den grp-IDs 5, 8, 11 und 12 sowie

zwei der Gruppen am oberen Bildrand. Eine der manuell Identifizierten Gruppen, wurde jedoch nicht vom System erkannt.

Abbildung 12 - Manuelle Identifikation von Gruppen nach 3 Sekunden


Abbildung 13 - Vom System ermittelte Gruppierungen nach 3 Sekunden

**Vergleich der Erkennung nach fünf Sekunden Videozeit**

Im Vergleich dieser Bilder (Abb. 14 und Abb. 15) ist zu erkennen, dass zwei der erwarteten Gruppen (grp-ID 1 und grp-ID 7) auch vom System erkannt wurden. Die anderen drei vom System erkannten Gruppen (grp-ID 8, grp-ID 15 und eine Gruppe am oberen Bildrand) wurden nicht manuell Identifiziert. Auch hier stimmten die Anzahl der jeweiligen Personen einer Gruppe meist nicht mit dem Erwartungswert überein.

Die manuell nach 3 Sekunden erneut markierte Gruppe mit der grp-ID 4 (vgl. grp-ID 4 Abb. 13) wurde vom System nach 5 Sekunden nicht wiedererkannt. Auf den Abbildungen ist zu erkennen, dass die Personen dieser Gruppe eine neue ID erhalten haben.

Außerdem ist zu sehen, dass die rechte Gruppe an der oberen Bildkante (vgl. Abb. 14) nicht als solche erkannt wurde. Stattdessen wurden im gleichen Bildbereich zwei andere Personen gruppiert.

Abbildung 14 - Manuelle Identifikation von Gruppen nach 5 Sekunden


Abbildung 15 - Vom System ermittelte Gruppierungen nach 5 Sekunden

Zusammenfassend zeigte sich, dass das entwickelte System Gruppen anhand definierter Bewegungs- und Gruppierungsparameter erkannt und markiert hat. So bestand zu den untersuchten Zeitpunkten immer eine Schnittmenge zwischen der Erwartungsmenge und der Menge aus den vom System erkannten Gruppen. Es wurden jedoch meist mehr Gruppen vom System erkannt als manuell identifiziert wurden. Im Vergleich war zu erkennen, dass die Anzahl von Personen innerhalb erkannter Gruppen ebenfalls von dem zugehörigen Erwartungswert abwich.

## 4.2 Szenenspezifische Auswertung der Gruppen

### 4.2.1 Hinzufügen von Personen zu bestehenden Gruppen

Um zu verdeutlichen, wie Personen durch das System zu bereits bestehenden Gruppen anhand der vorgegebenen Parameter hinzugefügt werden, wurde eine bestimmte Gruppe im zeiltichen Verlauf am oberen Bildrand herangezogen.

Zur besseren Erkennbarkeit wurden entsprechende Bildausschnitte vergrößert. Hierzu sei auf Abbildung 16 verwiesen: Abbildung 17 stellt den in rot verdeutlichten Ausschnitt vergrößert

dar. Abbildung 18 und Abbildung 19 sind entsprechend vergrößerte Ausschnitte eines Gesamtbildes zu den unter den Abbildungen genannten Zeitpunkten und stellen somit (zusammen mit Abb. 17) eine chronologische Abfolge dar. Die näher betrachtete Gruppe besteht aus vier Personen. Diese bewegen sich im Zeitverlauf in obere Richtung.

Abbildung 17 zeigt, dass zunächst zwei der vier Personen als Gruppe erkannt und markiert wurden. Die Person mit der ID 7 holt im zeitlichen Verlauf zur bestehenden Gruppe auf. Hierdurch wurden die Gruppierungsparameter erfüllt, folglich wurden sie zur Gruppe hinzugefügt – hier zu sehen in Abbildung 18.

Die Person mit der ID 13 hatte zunächst eine höhere Geschwindigkeit als die nun entstandene Gruppe und holte folglich auf. Auch hier wurden die Parameter erfüllt und eine Gruppe mit vier Personen entstand (vgl. Abb. 19).

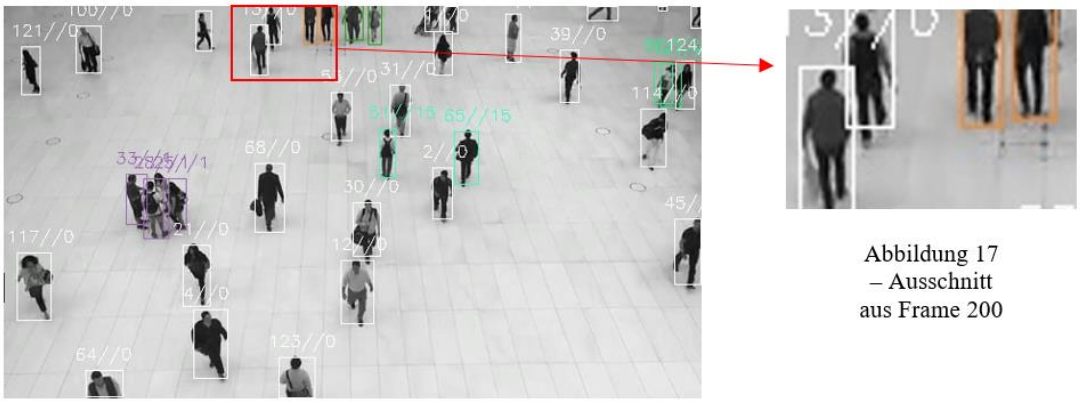


Abbildung 17 – Ausschnitt aus Frame 200

Abbildung 16 – Frame 200

Abbildung 18 – Ausschnitt aus Frame 210

Abbildung 19 – Ausschnitt aus Frame 220

### 4.2.2 Auflösen von Gruppen

In der Bildfolge – beschrieben durch Abbildung 20 und Abbildung 21 – ist zu erkennen wie sich die Gruppe mit der Gruppen-ID 15 auflöste. So erfüllten die zugehörigen Personen in Abbildung 20 noch die Gruppierungsparameter. Der Abstand dieser Personen mit den IDs 51 und 65 vergrößerte sich im Verlauf des Videos jedoch, bis diese nicht mehr als Gruppe erkannt wurden.


Abbildung 20 - Gruppen aus Frame 210


Abbildung 21 - Gruppen aus Frame 220

## 4.3 Konsistenz der (Gruppen-)Identifikationsnummern

Da das *Tracking* nur auf Basis aktueller *Detections* der „Detector"-Komponente durchgeführt wird, ist der *Tracking*-Algorithmus sehr anfällig auf Fehler in denselben. Ein Beispiel hierfür wurde bereits in Abschnitt 4.1 genannt. Im Folgenden sind die Auswirkungen dieser fehlerhaften *Detections* aufgezeigt:

Im Verlauf des Ergebnisvideos war zu erkennen, dass Personen, die sich in der oberen Ecke des Bildes befinden, sich oft ändernde IDs aufwiesen. Beispielsweise hatte eine Person oben links im Bild zunächst die ID 18 (vgl. Abb. 11) und im späteren Verlauf dann die IDs 115 (vgl. Abb. 22) und 121 (vgl. Abb. 20). Als weiteres Beispiel sei eine Person oben rechts im Bild angeführt: Sie erhielt zunächst die ID 16 (vgl. Abb. 15) und im späteren Verlauf dann die IDs 96 (vgl. Abb. 22) und 107 (vgl. Abb. 21).

Dieses Verhalten erklärt auch die Vergabe von schnell ansteigenden grp-IDs. Gruppen, die in diesen Bildbereichen erkannt wurden, gingen oftmals verloren und wurden dann wiedergefunden. Zusätzlich wurden die Personen aus der markierten Gruppe aus Abbildung 23 aufgrund fehlender und unzureichender *Detections* oft nicht *getrackt* und gruppiert. Auch hier erhielten

die zugehörigen Personen oft neue IDs und grp-IDs, was zu einem schnellen Anstieg derselben führte.

Im Gegensatz dazu wurden Personen und Gruppen, welche sich außerhalb der genannten Bereiche befinden, oftmals über ihre ganze Präsenzzeit hinweg *getrackt*. Die zugehörigen IDs und grp-IDs blieben entsprechend erhalten. Dies war an der Person mit der ID 12 (vgl. Abb. 11 und Abb. 21) und den Personen aus der Gruppe mit der grp-ID 1 (vgl. Abb. 11 und Abb. 21) gut zu erkennen. Die Personen in Gruppe mit der grp-ID 1 bewegten sich dabei kaum. Die Person mit der ID 12 bewegte sich im zeitlichen Verlauf von oben nach unten durch das Bild. Als weiteres Beispiel für eine Gruppe, welche im zeitlichen Verlauf erhalten blieb, ist die Gruppe mit der grp-ID 7 (vgl. Abb. 15 und Abb. 21 - Gruppe am oberen Bildrand links) zu nennen. Auch hier blieb die grp-ID erhalten während sich die Gruppe bewegte.


Abbildung 22 – Gruppen aus Frame 160

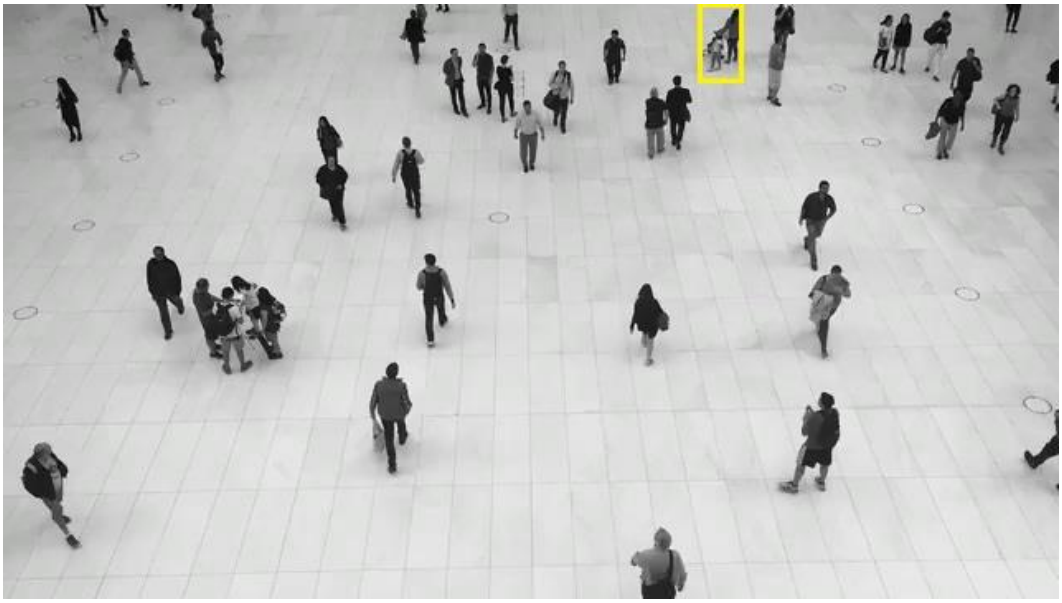


Abbildung 23 – Markierte Beispielgruppe

## 4.4 Abgleich der Ergebnisse mit den definierten Anforderungen

Es wurde gezeigt, dass die in Abschnitt 3.1 definierten Anforderungen an das System in Bezug auf das genutzte Beispielvideo erfüllt sind.

Das Video wurde vom System eingelesen. Personen wurden erkannt und erhielten über den Zeitraum, über den sie erkannt wurden, eindeutige IDs. Als neu interpretierte Personen erhielten hierbei eine neue, eindeutige ID. Die Personen wurden auf Basis ihrer errechneten Positionen und Durchschnittsgeschwindigkeiten über maximal eine Sekunde gruppiert. Wurde die Person noch nicht über mindestens eine Sekunde *getrackt*, so wurde die Durchschnittsgeschwindigkeit über diesen kürzeren *tracking*-Zeitraum berechnet. Die Personen einer Gruppe erhielten eine entsprechend eindeutige grp-ID. Diese grp-ID einer Gruppe blieb über die Zeit hinweg, über die diese Gruppe erkannt wurde, erhalten. Personen, die keiner Gruppe zugeordnet waren, erhielten die grp-ID 0 und wurden in den Bildern entsprechend markiert. Aus den daraus resultieren Bildern, in denen die Gruppen entsprechend markiert wurden, erstellte das System zum Schluss ein Video.

Eine der Teilanforderungen konnte im Ergebnisvideo nicht überprüft werden: So war im Video nicht zu beobachten wie eine Gruppe sich in mehrere Teilgruppen trennt. In diesem Fall hätte eine verbliebene Teilgruppe die alte grp-ID behalten sollen. Der anderen hätte eine neue grp-ID zugewiesen werden sollen.

# 5 Diskussion

## 5.1 Fazit

Das hier entwickelte System zur Gruppierung von Personen anhand ihrer Positions- und Bewegungsdaten beruht auf *Detections* von YOLOv3 (Abschnitt 2.1.3) und einem eigens entwickelten Algorithmus zum *Tracking* von Personen auf Basis ihrer jeweilig zuletzt erkannten Position.

Um einen Algorithmus zur Identifikation sozialer Gruppen in Videos zu erstellen, wurde zunächst YOLO als geeignete Methode zur Multi-Object-Detection ausgewählt. Da dieser Algorithmus als Basis für die weitere Arbeit genutzt wurde, erfolgte eine detaillierte Auseinandersetzung mit Entwicklung und Funktion dieser Methodik. Es zeigte sich, dass YOLOv3 für die Zielsetzung dieser Arbeit geeignet ist (Abschnitt 2.1.3).

Um Personen in Videos nicht nur erkennen zu können, sondern vielmehr auch zu Gruppen zuordnen zu können, wurde der NMSC-Algorithmus als Basis herangezogen. Wie beschrieben, ermittelt der Algorithmus ähnliche Gruppen wie die von Menschen identifizierten Gruppen. Aus diesem Grund wurden grundlegende Konzepte aus dem NMSC-Algorithmus für das hier entwickelte System übernommen.

Zur Lösung der Aufgabenstellung (Abschnitt 1.2) wurden zunächst folgende vier Kernprobleme identifiziert:

Erstens, müssen Personen in Bildern vom System erkannt und zweitens, auch in aufeinanderfolgenden Bildern wiedererkannt werden. Als Drittes müssen für alle Personen Positionen und Geschwindigkeiten berechnet werden. Zuletzt müssen Personen anhand dieser Daten gruppiert werden.

Daraus ergeben sich für die Entwiclung des Systems die drei Kern-Komponenten aus Abschnitt 3.2: „Detector“, „Tracker“, „Grouping“. Hierbei ist die „Grouping“-Komponente sowohl für die Berechnung der erforderlichen Positions- und Bewegungsdaten als auch für das Gruppieren der Personen verantwortlich. Um diese Daten berechnen zu können, wurde eine Komponente „Tracker“ entwickelt, welche sich für die Reidentifikation der Personen verantwortlich zeigt (vgl. Abschnitt 3.2.2).

In der Auswertung zeigte sich, dass der entwickelte *Grouping*-Algorithmus in der Lage ist, die identifizierten Anforderungen an das System zu erfüllen (vgl. Abschnitt 4.4): So werden Personen im Verlauf des Videos erkannt und wiedererkannt. Die Personen werden über eindeutige IDs im Ergebnisvideo (Anhang 3) gekennzeichnet. Zusätzlich wurde gezeigt, dass das System den Personen entsprechende grp-IDs zuweist, wenn diese die Gruppierungsparameter erfüllen. Bei Verlassen einer Gruppe erhält die betreffende Person wieder die grp-ID 0 (wenn sie nun keiner anderen Gruppe mehr zugeordnet ist). Der Fall, das eine Gruppe sich in mehrere Teilgruppen aufteilt, trat im Ergebnisvideo nicht auf. So konnte die entsprechende Anforderung im Ergebnisvideo nicht überprüft werden. Wünschenswert wäre diesbezüglich die Ergänzung eines geeigneten Testfalles, um die entsprechende Anforderung zu überprüfen.

In der Gegenüberstellung von manueller Gruppenidentifikation zu den vom System erkannten Gruppen zeigte sich, dass eine große Schnittmenge zwischen durch das System erkannten und manuell erkannten Gruppen bestand. In einigen Fällen erkannte das System jedoch aufgrund der gewählten Gruppierungsparameter mehr Gruppen, als manuell identifiziert wurden (vgl. Abschnitt 4.1). Das scheint vor allem darin begründet zu sein, dass auch in kurzem Abstand hintereinander gehende Personen, von dem System als Gruppe erkannt wurden. Im Gegensatz dazu scheinen die Kriterien für das Bestehen einer Gruppe aus menschlicher Sicht enger gewählt.

Außerdem zeigte sich, dass fehlerhafte *Detections* einen großen Einfluss auf den hier entwickelten *Tracking*-Algorithmus haben. Das wiederum führt dazu, dass nicht *getrackte* Personen keiner Gruppe hinzugefügt werden können (vgl. Abschnitt 4.3).

## 5.2 Einschränkungen des entwickelten Systems

Im Gruppierungsvorgang bestand eine starke Abhängigkeit zu den gewählten *Detection*- und *Tracking*-Methoden (vgl. Abschnitt 4.3):

Der hier entwickelte Algorithmus zum *Tracking* von Personen nutzte lediglich die zuletzt bekannte Position einer Person, um eine entsprechende Zuordnung einer *Detection* zu dieser vorzunehmen. Wenn eine Person also im zeitlichen Verlauf, aus beliebigen Gründen (fehlende *Detections*, Person verschwand hinter einem Objekt) nicht erkannt wurde, so wurde sie vom System danach als neue Person interpretiert. Das hatte zur Folge, dass diese als neu interpretierte Person einer erst mit einem zeitlichen Versatz wieder einer Gruppe zugeordnet werden konnte. Das System war also nicht in der Lage, Personen zu reidentifizieren, welche über einen Zeitraum von mehr als vier Frames nicht mehr erkannt wurden (vgl. Abschnitt 3.2.2).

Der hier entwickelte *Tracking*-Algorithmus funktionierte entsprechend im ausgewählten Beispielvideo. Aufgrund der ermittelten Grenze, wie weit sich ein Zentrum zweier *Detections* einer Person zwischen zwei Frames bewegen durfte, ist dieser vermutlich jedoch nicht allgemeingültig funktionsfähig. So würden Personen, deren erkanntes Zentrum sich zu schnell verschoben hat, nicht reidentifiziert werden.

Hervorzuheben ist auch die Relevanz bezüglich der Definition einer sozialen Gruppe: So beruhte die Definition in dieser Arbeit alleinig auf der Position und Geschwindigkeit der Personen und berücksichtigte unter anderem nicht ihre Anordnung zueinander. Denkbar wäre hier beispielsweise eine Unterscheidung zwischen voreinander und nebeneinander gehenden Personen. Auch ähnliche Erscheinungsbilder (z.B. hervorgerufen durch Uniformen) von Personen oder etwaige interpretierbare soziale Konstellationen zwischen diesen (bspw. Eltern–Kind Beziehung) könnten in der Weiterentwicklung Berücksichtigung finden. Zuletzt wäre auch der Einbezug von Interaktionen denkbar: Miteinander sprechende oder durch Gesten kommunizierende Personen könnten gegebenenfalls auch sozialen Gruppen zugeordnet werden.

Bei Anwendung der hier genutzten Methoden zur Multi-Object-Detection und zum Multi-Object-Tracking war weiterhin auf eine geeignete Parametrisierung zu achten: Besonders wichtig hierbei schienen die Parameter für die angewandte *Non-Maxima-Supression* im „Detector“, sowie der maximal erlaubte Abstand zwischen Zentren derselben Person in aufeinanderfolgenden Frames. Dies funktionierte nur, solange die Anzahl der aufgenommenen Bilder pro Sekunde so hoch ist, dass eine Person im zeitlichen Abstand zwischen zwei Frames nicht die Position einer anderen Person einnehmen konnte.

## 5.3 Ausblick

Abhängig vom gewählten Anwendungsszenario könnte es durchaus eine Rolle spielen, Personen zu reidentifizieren, welche über einen längeren Zeitraum nicht erkannt wurden: Ist dies der Fall, so wäre ein *Tracking*-Ansatz zu wählen, welcher das Erscheinungsbild einer Person berücksichtigt; beispielsweise präsentierten Wojke et al. (2017) mit *DeepSORT* einen solchen. Auch Chahayati et al. (2017), verglichen zwei *Tracking*-Ansätze, welche auf dem Erscheinungsbild der erkannten Personen beruhen. Sie beschrieben, dass die Nutzung des *Faster R-CNN* von Ren et al. (2016) sich besonders gut dafür eignet, da die vom CNN ermittelten Features pro Person direkt untereinander Vergleichbar wären (vgl. Chahayati et al., 2017).

Die vorhandenen Einschränkungen der hier entwickelten Grouping-Methode bestanden primär aus fehlerhaften *Detections* und fehlerhaftem *Trackings* (vgl. Abschnitt 4.3). So wurde der *Grouping*-Algorithmus unter Anwendung geeigneter Methoden (vgl. Abschnitt 3.2.3) zwar dahingehend optimiert, dass der Einfluss fehlerhaftem *Trackings* minimiert war, allerdings würden verbesserte Multi-Object-Tracking Methoden vermutlich auch zu einer äquivalenten Verbesserung des Gesamtsystems führen.

Die Homogenisierung des Koordinatensystems und die daraus resultierenden berechneten Werte für Position und Geschwindigkeit berücksichtigten hier nicht die Höhe einer Person. Stattdessen wurde das Ziel-Koordinatensystem auf Basis einer Referenzfläche auf dem Boden bestimmt. Aufgrund dessen spielte die Genauigkeit - vor allem die, der erkannten Dimensionen von Objekten – der gewählten Multi-Object-Detection Methode für den entwickelten *Grouping*-Algorithmus eine wichtige Rolle. So wurde die Position einer Person in diesem Algorithmus als Mitte derjenigen Seite einer *Bounding-Box* interpretiert, an der sich die Füße der Person

befinden. Daraus resultierten Ungenauigkeiten in den Positions- und Bewegungsdaten. Die Auswirkungen dieses Rauschens wurden zwar wie beschrieben minimiert, eine verbesserte Methode zur Ermittlung dieser Daten, ggf. auf Basis des sich bewegenden Zentrums einer Person, könnte jedoch auch zu einer äquivalenten Verbesserung in der Erkennung von Gruppen führen.

# Literaturverzeichnis

Berry, C. C. (1992). The kappa statistic. *JAMA*, *268* (18), 2513–2514. doi:10.1001/jama.268.18.2513

Bhargava, N. & Chaudhuri, S. (2014). Finding group interactions in social gathering videos. *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing* (S. 1–8). Gehalten auf der ICVGIP ’14: Indian Conference on Computer Vision Graphics and Image Processing, Bangalore India: ACM. doi:10.1145/2683483.2683488

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*.

Chandran, A. K., Poh, L. A. & Vadakkepat, P. (2015). Identifying social groups in pedestrian crowd videos. *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)* (S. 1–6). Gehalten auf der 2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR), Kolkata, India: IEEE. doi:10.1109/ICAPR.2015.7050677

Cheng, Z., Qin, L., Huang, Q., Jiang, S., Yan, S. & Tian, Q. (2011). Human group activity analysis with fusion of motion and appearance information. *Proceedings of the 19th ACM international conference on Multimedia* (S. 1401–1404). Gehalten auf der MM ’11: ACM Multimedia Conference, Scottsdale Arizona USA: ACM. doi:10.1145/2072298.2072025

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. Verfügbar unter: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html [Zuletzt aufgerufen am 13.07.2023]

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. Verfügbar unter:

http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html [Zuletzt aufgerufen am 13.07.2023]

Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A. & Berg, A. C. (2017). DSSD: Deconvolutional Single Shot Detector. arXiv. Verfügbar unter: http://arxiv.org/abs/1701.06659 [Zuletzt aufgerufen am 13.07.2023]

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv. Verfügbar unter: http://arxiv.org/abs/1512.03385 [Zuletzt aufgerufen am 13.07.2023]

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv. Verfügbar unter: http://arxiv.org/abs/1502.03167 [Zuletzt aufgerufen am 13.07.2023]

Koech, K. E., (2020). Object Detection Metrics With Worked Example. Verfügbar unter: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e [Zuletzt aufgerufen am 13.07.2023]

Lin, M., Chen, Q. & Yan, S. (2014). Network In Network. arXiv. Verfügbar unter: http://arxiv.org/abs/1312.4400 [Zuletzt aufgerufen am 13.07.2023]

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. et al. (2016). SSD: Single Shot MultiBox Detector (Band 9905, S. 21–37). doi:10.1007/978-3-319-46448-0_2

Maheshwari, S. & Heda, S. (2016). A Review on Crowd Behavior Analysis Methods for Video Surveillance. *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (S. 1–5). Gehalten auf der ICTCS '16: Second International Conference on Information and Communication Technology for Competitive Strategies, Udaipur India: ACM. doi:10.1145/2905055.2905258

*People Commerce Shop free stock video*. (2016). Verfügbar unter: https://pixabay.com/videos/people-commerce-shop-busy-mall-6387/ [Zuletzt aufgerufen am 13.07.2023]

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv.Verfügbar unter: http://arxiv.org/abs/1506.02640 [Zuletzt aufgerufen am 13.07.2023]

Redmon, J. & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv. Verfügbar unter: http://arxiv.org/abs/1612.08242 [Zuletzt aufgerufen am 13.07.2023]

Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv. Verfügbar unter: http://arxiv.org/abs/1804.02767 [Zuletzt aufgerufen am 13.07.2023]

Ren, S., He, K., Girshick, R. & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv. Verfügbar unter: http://arxiv.org/abs/1506.01497 [Zuletzt aufgerufen am 13.07.2023]

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. et al. (2015). ImageNet Large Scale Visual Recognition Challenge. arXiv. Verfügbar unter: http://arxiv.org/abs/1409.0575 [Zuletzt aufgerufen am 13.07.2023]

Shetty, A. K., Saha, I., Sanghvi, R. M., Save, S. A. & Patel, Y. J. (2021). A Review: Object Detection Models. *2021 6th International Conference for Convergence in Technology (I2CT)* (S. 1–8). Gehalten auf der 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India: IEEE. doi:10.1109/I2CT51068.2021.9417895

Terven, J. & Cordova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 and Beyond. arXiv. Verfügbar unter: http://arxiv.org/abs/2304.00501 [Zuletzt aufgerufen am 13.07.2023]

Wirz, M., Schläpfer, P., Kjærgaard, M. B., Roggen, D., Feese, S. & Tröster, G. (2011). Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of GPS trajectories. *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks* (S. 17–24). Gehalten auf der GIS '11: 19th SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago Illinois: ACM. doi:10.1145/2063212.2063220

# A Anhang

Alle Anhänge werden in digitaler Form eingereicht.

## A.1 Anhang 1: Beispielvideo

Zu finden unter: <\Bachelorarbeit Tobias Ranfft abgegeben am 14.07.23\Anhang A.1 – Beispielvideo>

## A.2 Anhang 2: System

Zu finden unter: <\Bachelorarbeit Tobias Ranfft abgegeben am 14.07.23\Anhang A.2 – System> (*People Commerce Shop free stock video*, 2016)

## A.3 Anhang 3: Ergebnisvideo

Zu finden unter: <\Bachelorarbeit Tobias Ranfft abgegeben am 14.07.23\Anhang A.3 – Ergebnisvideo>

**Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit**

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

| _____ | _____ | _____ |
|---|---|---|
| Ort | Datum | Unterschrift im Original |