# Videoanalyse und Objekttracking

1st Bartolovic Eduard
*Hochschule München*
München, Deutschland
eduard.bartolovic0@hm.edu

2nd Thomas Willeit
*Hochschule München*
München, Deutschland
XXXXX@hm.edu

3rd Schäfer Julia
*Hochschule München*
München, Deutschland
j.schaefer0@hm.edu

***Zusammenfassung*—**

## I. Konzept

Ziel des Projektes ist es klassische Verfahren mit neuere DeepLearning Verfahren zu vergleichen. Dafür ist für beide arten eine Pipeline geschaffen worden.
Für die Klassische Verfahren wurde für die Objekterkennung Gausmixture verwendet und für das Tracking Sort. Für die DeepLearnig Verfahren wurde für die Objekterkennung ein Neuronales Netz und für das Tracking wurde Deepsort verwendet.

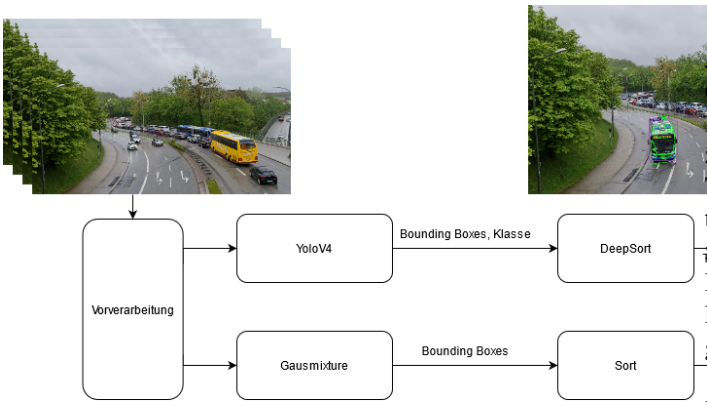


Abbildung 1. Konzept für das Projekt

## II. Vorverarbeitung

### A. Rauschminderung

## III. Objekterkennung: Gauss Mixture

## IV. Objekterkennung: YoloV4

YOLO ist ein Neuronal Netz für die Echtzeit-Objekterkennung. YOLO ist die Abkürzung für 'You Only Look Once' was übersetzt 'Man sieht nur einmal hin' heißt. Die Aufgabe der Objekterkennung besteht darin, den Ort und die Bounding Box im Bild zu bestimmen, sowie die Objekte zu klassifizieren. Frühere Methoden, wie R-CNN und seine Variationen, verwendeten eine Pipeline, um diese Aufgabe in mehreren Schritten durchzuführen. Dies ist in der Ausführung langsam und aufwendiger zu trainieren, da jede einzelne Komponente separat trainiert werden muss. YOLO, erledigt beide Aufgaben mit einem einzigen Convolutional Neural Network. Es betrachtet dabei die Objekterkennung als ein Regressionsproblem auf räumlich getrennte Bounding Boxes und zugehörige Klassenwahrscheinlichkeiten. [1]
Das Bild in ein S × S-Gitter mit den Residualblöcken aufgeteilt. Wenn der Mittelpunkt eines Objekts in eine Gitterzelle fällt, ist diese Gitterzelle für die Erkennung dieses Objekts zuständig. Jede Gitterzelle sagt B Bounding Boxes und C Klassen Konfidenzwerte für diese Boxen voraus [1]
Diese Klassen Konfidenzwerte zeigen, wie sicher das Modell ist, dass die Boundingbox ein Objekt enthält und auch wie genau die Box das Objekt beschreibt. Die Konfidenz wird wie folgt definiert:

$$c = Pr(Objekt) * IOU_{pred}^{truth}$$

Jede Boundingbox enthält die 5 Vorhersagen: x,y,w,h,c. Jede Gitterzelle sagt auch bedingte Klassenwahrscheinlichkeiten $Pr(Class_i|Object)$, voraus. Diese Wahrscheinlichkeiten beziehen sich auf die Gitterzelle, die ein Objekt enthält. Zum Testzeitpunkt wird die bedingten Klassenwahrscheinlichkeiten und die individuellen Box-Vertrauensvorhersagen multipliziert, wodurch wir klassenspezifische Vertrauenswerte für jede Box erhalten. Diese Werte kodieren sowohl die Wahrscheinlichkeit, dass diese Klasse in der Box vorkommt, als auch, wie gut die vorhergesagte Box zum Objekt passt.

$$Pr(Class_i|Object) * Pr(Object) * IOU_{truth}^{pred} = Pr(Class_i) * IOU_{truth}^{pred}$$

.....Netzwerk???..... .....LossFunction???.....

In YoloV2 wurden Ankerboxen eingeführt. Da Objekte meist immer ähnliche Formen haben kann man eine gewisse Menge an sogenannten Ankerboxen definieren welche als Basis Boundingboxen fingieren. So wird anstatt die absolute Größen von Boxen in Bezug auf das gesamte Bild vorherzusagen, eine Ankerbox ausgewählt die am besten zu dem gewünschten Objekten passende verwendet. Diese Ankerboxen kann man mittels Clusteralgorithmen wie K-Means und den Boundingboxen aus dem Datensatz berechnen werden. Dies hat vor allem den Vorteil das mehr Boundingboxen vorhersagbar sind als noch zuvor. [3]
..... YoloV3?? YoloV4?? .... In diesem Projekt verwendeten wir die 4te und damit aktuell neueste Version von Yolo [2]. So bietet jede Version inkrementelle Verbesserung zum jeweiligen Vorgänger.
Unser Modell wurde bereits mit dem Microsoft COCO Datensatz trainiert. Es sind 80 verschiedene Klassen erkennbar. Wir interessieren uns aber nur für einen kleineren Teil wie:

1) Personen

2) Pkw
3) Lkw
4) Busse
5) Fahrräder
6) Züge

Je nach Szenario lassen sich per Parameter die relevanten Klassen auswählen. Das Modell prädiktiert trotzdem noch alle Klassen. Die nicht relevanten werden einfach im Postprocessing herausgefiltert. Das Ergebnis einer Prädiktion ist eine Liste von Objekten welches aus den x,y Koordinate des Mittelpunktes der Boundingbox, der Breite w, der Höhe h, der Klasse und des Konfidenzwerts. Ein weitere Postprocesingschritt ist die Non-Maxima-Suppression. BEISPIELBILD? Der Output kann überschneidende Boundingboxen besitzen die eigentlich zur selben Klasse gehören. Diese Duplikate können mittels der Non-Maxima-Suppression entfernt werden. Heirfür wird die IOU der Boxen berechnet. Sollte die IOU einen Threshold überschreiten dann wird das Duplikat entfernt.

### A. Fehler in der Erkennung

...falsch Klassifizierung wegen schlechter Datensatz.... Zu großer Datensatz...

YOLO besitzt starke räumliche Beschränkungen für die Boundingbox-Vorhersagen, da es pro Gitterzelle nur eine begrenzte Anzahl an Vorhersagen treffen kann. Diese Anzahl ist direkt Abhängig von der Anzahl der Ankerboxen [3]. Diese räumliche Beschränkung begrenzt die Anzahl der Objekte in der Nähe, die das Modell vorhersagen kann. ————- Problem zu viele Elemente nebeneinander..... wegen zu großen matschigen Grid zu wenig Ankerboxen????————

## V. Tracking: Sort

SORT berechnet auf Basis des Kalman Filters und der Hungarian Methode die Bewegung des Objekts und weist je nach Status eine eindeutige Identifikationsnummer zu. Der Kalman-Filter funktioniert für die Problemstellung sehr gut, da dieser ein lineares System mit gaußisches Rauschen voraussetzt. Dies ist gegeben durch die gleichmäßige Bewegung der Züge/ Autos in eine Richtung. Der Kalmanfilter kann aus verrauschten, teils redundanten Messungen die Zustände und Parameter des aktuellen Systems schätzen. Die Zustände des aktuellen Systems sind die Positionen angegeben in Bounding Boxen der Objekte. Durch Berücksichtigung des letzten Zustand des Systems, (die letzte Position), der Schätzung der nächsten Position und Abgleich mit der aktuellen Messung (die aktuelle Detektion) und anschließender Aktualisierung des Systems, kann eine zuverlässige Verfolgung des Objekts berechnet werden. Probleme die den Kalmanfilter stören sind zum Beispiel ungleichmäßige Bewegungen von Fahrzeugen, dies wird „Process Noise“ genannt, sowie Ungenauigkeiten in Messungen, diese werden als „Measurement Noise“ bezeichnet. Kann für zu dem aktuell getrackten Objekt in dem Frame keine Bounding Box zugeordnet werden, wird der Zustand des Systems auch ohne die Korrektur durch die Messung berechnet und kann verwendet werden, sollte in einem späteren Frame wieder eine Bounding Box zugeordnet werden können. Die Hungarian Methode wird verwendet um die Detektion einem existierenden Objekt zuzuorden. Hierfür wird die Intersection-over-Union Distance zwischen jeder Detektion und allen bereits erkannten Objekten verglichen. Die Assoziationsmatrix kann optimal durch die Hungarian Methode berechnet werden. Als Verbesserung wurde hier bei der Implementierung durch DeepSort noch eine Mindestüberschneidung $IoU_{min}$ eingeführt.

## VI. Tracking: DeepSort

SORT an sich ist bereits ein Tracker, das Problem ist jedoch, dass die ID-Änderungen zu oft auftreten. Das heißt die Objekt-ID wechselt während der Verfolgung. Ausgelöst wird dies zum Beispiel durch die teilweise Verdeckung des Objekts. Deshalb wurde SORT um ein CNN erweitert, welches sowohl einerseits die Mahalanobis-Distanzmetrik für die Assoziation nutzt, als auch die Form des Objekts in Form eines durch das hinzugefügte CNN berechneten Feature-Vektor verfolgt. Durch Verwendung einer geeigneteren Distanzmetrik und Beschreibung des Objekts kann dieses nun besser verfolgt werden. Das CNN erstellt einen Vektor, der enthält alle Features eines gegebenen Bilds. Ursprünglich wird das Neuronale Netz als Klassifikator trainiert. Indem die Klassifikations Layer weggelassen werden, können die resultierenden Feature-Vektoren als Wiedererkennungsform verwendet werden.

Anstatt bei der Ungarischen Methode wie bei SORT die IoU-Distanz zu verwenden, wird nun die (quadrierte) Mahalanobis Distanz verwendet, welche die Distanz zwischen der detektierten Bounding Box und der gaußverteilten Vorsage des Zustands des Kalman-Filters berechnet:

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \tag{1}$$

Hierbei wird die Verteilung, zuvor acht-dimensional des i-ten verfolgen Objekts in den vier-dimensionalen Raum der Messungen projiziert $(y_i, S_i)$. Diese Bounding Box der Detektion wird mit $d_j$ bezeichnet.

| | |
|---|---|
| Kalman-Filter Zustand: | $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ |
| Detektion: | $(u, v, \gamma, h)$ |
| $u, v$ : | Mittelpunkt der Bounding Box |
| $h$ : | Skalierungsfaktor |
| $\dot{x}, \dot{y}, \dot{\gamma}, \dot{h}$: | Geschwindigkeiten in Bildkoordinaten |

Die Mahalanobis Distanz berechnet den Abstand von einem Punkt zu einer Verteilung. Der Abstand ist dabei wie viele Standardabweichungen der Verteilung dieser Punkt vom Mittelwert der Verteilung entfernt ist. Er wird somit durch die Standardabweichung der Verteilung normiert. Die Detektion stellt hierbei den Punkt da und die Unsicherheit der Zustandsvorhersage, beziehungsweise die Unsicherheit der Position des getrackten Objekts, die Verteilung. Dadurch wird bei der Berechnung der Distanz die Unsicherheit der Position miteinbezogen.

Für die Distanzmetrik ergibt sich dann folgende Gleichung $D = \lambda * D_k + (1 - \lambda) * D_a$. Hierbei ist $D_k$ die Mahalanobis

Distanz und $D_a$ die kleinste Kosinus-Distanz zwischen den Feature-Vektoren. $\lambda$ dient als ein Gewichtungsfaktor.

Kann DeepSORT erfolgreich ein bereits verfolgtes Objekt zu einer Detektion assoziieren, wird anschließend eine TrackingID erstellt. Hierbei muss beachtet werden, dass im Zuge der Verbesserung von SORT weitere Sicherheiten eingebaut wurden. Ein neu getracktes Objekt erhält nicht sofort eine endgültige Identifikationsnummer. Erst nachdem das Objekt in drei hintereinander liegenden Frames erfolgreich getrackt werden konnte erhält es eine Nummer. Das Objekt erhält des weiteren einen Zähler $a$, der die Anzahl an Frames zählt, seitdem das Objekt nicht mehr erfolgreich assoziiert werden konnte. Ist eine maximale Anzahl $Age_{max}$ erreicht gilt das Objekt als verloren oder außer Reichweite der Kamera und wird aus der Objektliste entfernt. Dieser Zähler wird jedoch zurückgesetzt, sollte zuvor das Objekt erneut erfolgreich assoziiert werden.

## VII. Zählen von Objekten

## VIII. Zustandserkennung Züge

Da nun ein Objekt erfolgreich erkannt, verfolgt und seine Bewegungsrichtung erkannt werden kann, wird nun für Objekte die die Klasse Zug haben die aktuelle Zugphase bestimmt. Durch YOLO erhält man die Bounding Box. Durch abgleich der TrackingID kann festgestellt werden, ob es ich auch immer um den gleichen Zug handelt. Für jede dessen Detektionen wird durch Optical Flow die Magnitude und die Richtung bestimmt. Fährt der Zug ein, hat dieses Objekt eine Magnitude größer 1, die durchschnittliche Richtung wird über die gesamten Richtungsvektoren der Bounding Box des Zuges bestimmt und als Richtungspfeil in das Bild eingefügt. Sein Zustand wechselt von „Kein Zug“ zu „Einfahrend“. Dabei wird immer für 5 Frames geprüft ob die Richtung gleichbleibend ist. Sollte für eine Länge von 10 Frames die Magnitude unter 0 fallen ändert sich der Zustand des Zuges zu „Haltend“. Beginnt der Zug sich wieder zu bewegen und die Richtung ist wieder gleichbleibend in eine Richtung wird der Zustand auf „Abfahrend“ gesetzt, bis der Zug außerhalb des Bildes ist und nicht mehr erkannt wird.

## Literatur


[1] https://arxiv.org/pdf/1506.02640v5.pdf
[2] https://arxiv.org/pdf/2004.10934.pdf
[3] https://arxiv.org/pdf/1612.08242.pdf


## IX. Vergleich klassischen Methoden mit DeepLearning

## X. Anhang