

1 Global Nearest Neighbour: Algorithmus und Idee

2 Grundlage GNN

Da die Verfolgung der exakten gesuchten posteriori Wahrscheinlichkeit, aufgrund begrenzter Rechnerzeit und hohen Rechneraufwand, nicht genau bestimmbar ist, sind Approximationen für eine durchführbare Rechnung erforderlich. In diesem Kapitel wird der einfachste und rechnen günstigste Tracking Algorithmus vorgestellt, der Global Nearest Neighbour (*GNN*).

Wie bereits erwähnt, werden hauptsächlich zwei Annahmen getroffen, um die Problematik noch berechenbar zu gestalten, die sogenannte Merging und Pruning. Der *GNN* basiert sich auf ein abruptes Pruning.

2.1 Bekannte Anzahl von Objekten

Eine wichtige Annahme und gleichzeitig Einschränkung dieses Algorithmus, bezieht sich auf die Voraussetzung, dass die Anzahl der Objekte in jedem Zeitschritt n_k bekannt ist. Diese Annahme ist hinsichtlich einer Performance Analyse auf künstliche Testdaten ausreichend, allerdings wird der Algorithmus für eine Realitätsnahe Simulation so gut wie unbrauchbar. In den späteren Kapiteln, wird die *M/N* Methode vorgestellt, die eine Einschätzung von n_k liefern kann, daher wäre der GNN bei einer praktischen Anwendung verwendbar.

2.2 Pruning in dem GNN Context

Hinsichtlich der Übersichtlichkeit der vorhanden Problematik, stellt man sich folgendes Szenario vor: Bei jedem Zeitschritt k (immer wenn ein neues Bild aufgenommen wird), erhält der Algorithmus verschiedenen Messungen \underline{M}_k mit x und y Koordinaten und Länge m_k aus der Objektdetektion. Dies enthält sowohl Clutters wie die Koordinaten der echten Objekten o_k . Ohne eine genauere Analyse, sind diese beiden Größen nicht unterscheidbar, daher bewertet man über Zeitschritten hinweg, welche der Koordinaten näherungsweise einem *Constant Velocity Model* entsprechen können. Je mehr die Reihe der Koordinaten einer konstanten Geschwindigkeit Annahme folgt, desto wahrscheinlicher ist es, dass es sich um ein echtes Objekt handelt. Auf der anderen Seite, Reihen von Koordinaten deren vorkommen zufällig erscheinen, werden mit höherer Wahrscheinlichkeit den Clutters zugewiesen.

Eine Datenanalyse ohne das Pruning, müsste alle mögliche Kombinationen der Daten bewerten. Zum Beispiel für lediglich zwei Zeitschritten $k = 0$ und $k = 1$ mit jeweiligen zwei Messungen $m_0 = 2$ und $m_1 = 2$, entstehen 4 Assoziationsmöglichkeiten, da sowohl die erste, wie die zweite Koordinate von der Messung M_0 zwei Alternativen besitzen sich mit den Koordinaten der M_1 zu Assoziieren.

Laut (of Technology, 2019) die Gleichung, die die gesamte Anzahl aller Assoziationen N_A für beliebige Objektanzahl n_k und beliebige Messungen m_k vorgibt

lautet:

$$N_A(m_k, n_k) = \sum_{\delta=0}^{\min(m_k, n_k)} \frac{m_k! n_k!}{\delta! (m_k - \delta)! (n_k - \delta)!} \quad (1)$$

Infolge dessen, für $m_k = n_k = 8$, entstehen bereits 1441729 mögliche Assoziationen. Dieses Beispiel vermittelt wie wesentlich für den Rechneraufwand es sei, die Hypothesen Anzahl mithilfe des Prunings einzuschränken.

Bei dem *GNN*, wird lediglich die wahrscheinlichste Assoziation beibehalten, alle anderen möglichen Kombinationen werden bei jedem Zeitschritt *weggeprunt*. Diese Eigenschaft verleiht dem *GNN* die Schnelligkeit und Einfachheit, die seine Anwendung auf Kosten der Genauigkeit gerechtfertigt. Aus dem Grund, dass nur die wahrscheinlichste Hypothese bei jedem Zeitschritt k berücksichtigt wird, wurde dieser Algorithmus als *Greedy-Algorithm* bezeichnet.

Mithilfe des Prunings, kann man die folgenden posteriori Wahrscheinlichkeit der Objektzuständen \underline{x}_k darstellen.

$$p_{k|k}^{GNN}(\underline{x}_k) = p_{k|k}^{\theta_{1:k}^*}(\underline{x}_k), \quad (2)$$

wo θ_k^* die wahrscheinlichste Assoziationshypothese repräsentiert. $\theta_{1:k}^*$ stellt eine Reihe von wahrscheinlichsten Assoziationshypothesen bis den Zeitschritt k dar. Das Ziel wäre natürlich die posteriori Wahrscheinlichkeit für die jeweilige Objekte getrennt auszurechnen, also alle $p_{k|k}^{\theta_{1:k}^*, i}(\underline{x}_k^i)$ für $i = 1, \dots, n_k$.

2.3 Allgemeine Vorgehensweise

Mit einer bekannten Anzahl von Objekten n_k und ohne weitere Vereinfachungen sind die Hauptschritte dieses Algorithmus folgendermaßen abgebildet.

Für jeden Zeitschritt k :

Step 1: Prädiktion

Für jedes Objekt i , $p_{k|k-1}^i(\underline{x}_k^i)$ mithilfe einer Chapman-Kolmogorov Prädikation ausrechnen

Step 2: Wahrscheinlichste Hypothese ermitteln.

Step 3: Update

Für jedes Objekt i , $p_{k|k}^i(\underline{x}_k^i)$ mithilfe eines Bayes-Update ausrechnen, falls das Objekt detektiert worden ist. Ansonsten $p_{k|k}^i(\underline{x}_k^i) = (1 - P^D) p_{k|k-1}^i(\underline{x}_k^i)$. $P^D \in [0, 1]$ sei die Detektionsrate und wird als Konstant angenommen.

Step 4: Zustand aus der Aposteriori Wahrscheinlichkeit mithilfe des Erwartungswerts bestimmen

$$\underline{x}_k = \int \underline{x}_k^i p_{k|k}^{GNN}(\underline{x}_k) d\underline{x}_k^i$$

2.4 Unabhängige Objektzustände

Da die vorherigen Schritte für eine reale Anwendung unpraktisch sein können, setzt man folgenden Annahmen voraus.

Assumption 1 - Das Zustandsmodell sei linear.

Assumption 2 - Die Objektzustände seien linear unabhängig.

Assumption 3 - Gauß'sche Wahrscheinlichkeitsdichte der Objektbewegung.

Seien Sie erfüllt, ist der Einsatz eines Kalman-Filters bei der Prädikation beziehungsweise des Updates möglich, was die Berechnung wesentlich vereinfacht. Außerdem, kann man sich auf die Berechnung auf Step 4: verzichten, da eine direkte Ermittlung der Zustände ohne die Aposteriori Wahrscheinlichkeit möglich ist. Das nächste Kapitel zeigt wie der Algorithmus mithilfe dieser Vereinfachungen aufgebaut wird.

3 GNN Algorithmus

3.1 GNN Kalman-Filter

Wenn die Voraussetzungen 1 bis 3 des vorherigen Kapitels erfüllt sind, kann der GNN die Eigenschaften des Kalman-Filters ausnutzen. Dieses Filter handelt sich um einen Zustandsschätzer, der das Eingangs beziehungsweise Ausgangsrauschen eines dynamischen Systems minimieren kann. Dafür werden zwei Hauptschritten erforderlich: die Prädikation und die Korrektur (oder Update). Das Kalman-Filter benötigt drei Parameter, die die Leistung der Schätzung stark beeinflussen können. Die positiv-definite Matrix der Varianz des Modellrauschens $\mathbf{Q} \in \mathbb{R}^{4 \times 4}$, die positiv-definite Matrix der Varianz des Messrauschens $\mathbf{R} \in \mathbb{R}^{2 \times 2}$ und letztlich die symmetrische Objekt Kovarianzmatrix des Systemrauschens $\mathbf{P}^i \in \mathbb{R}^{2 \times 2}$. Die Matrizen \mathbf{Q} und \mathbf{R} sind für alle Objekte gleich und Konstant. Matrix \mathbf{P}^i wird bei jedem Korrekturschritt aktualisiert und ist für die jeweilige Objekte i unterschiedlich. Die Prädiktion und Korrektur werden bei jedem Zeitschritt erneut aufgerufen und sind für jedes Objekt $i = 1, \dots, n_k$ getrennt auszuführen. Der Einfluss der Wahl der Matrizen \mathbf{Q} und \mathbf{R} sowie des Startwertes der Matrix \mathbf{P}^i wird im Kapitel 1.3 untersucht.

3.2 Kalman-Prädiktion

Das Ziel der Prädiktion ist die Vorhersage des Zustands des aktuellen Zeitschrittes nur anhand des Modells (Matrizen \mathbf{F} und \mathbf{H}) und der Messung der Koordinaten bei dem vorherigen Schritt $k - 1$. In diesem Vorgang wird die Kenntnis der Größen des aktuellen Zeitschrittes noch nicht vorausgesetzt. Die Prädiktion erfolgt mithilfe folgender Gleichungen.

$$\underline{x}_{k|k-1}^i = \mathbf{F} \underline{x}_{k-1|k-1}^i, \quad (3)$$

$$\mathbf{P}_{k|k-1}^i = \mathbf{F} \mathbf{P}_{k-1|k-1}^i \mathbf{F}^T + \mathbf{Q}. \quad (4)$$

Die Notation $[\cdot]_{k|k-1}$ repräsentiert die Vorhersage des Zustands in dem Zeitschritt k gegeben die Informationen aus $k - 1$.

Die Prognose der aktuellen Werten ohne das Wissen der eigentlichen Größen verursacht eine Erhöhung der Fehleranfälligkeit, die ohne die Korrektur, die echte Ergebnis über viele Zeitschritten immer weiter verfälscht. Um dies zu vermindern, korrigiert der Update-schritt die vorhergesagte Größe bezüglich der gemessenen Werten.

3.3 Datenassoziation und Kostenmatrix

Bevor eine Korrektur auf die Prädiktion angewandt werden kann, wird die Ermittlung der wahrscheinlichsten Hypothese erforderlich. Das heißt: Bei einem Zeitschritt k , erhält man aus der Detektion, m_k Koordinaten und aus dem M/N Algorithmus, n_k Objektanzahl. Das Ziel ist natürlich die Objekte n_k einer der m_k Koordinaten zuzuweisen, um eine Korrektur mit den echten Messungen zu ermöglichen. Die Koordinaten die keinem Objekt entsprechen werden als Clutter angesehen. Außerdem, es besteht auch die Möglichkeit, dass ein bestimmtes Objekt nicht detektiert wurde. Um dies zu Lösen verwendet man die Eigenschaften eines *Zuweisungsproblems*.

Zur Veranschaulichung der Problematik, wird folgendes Beispiel präsentiert. An einem bestimmten k sei $n_k = 2$ und $m_k = 3$. Sei $l_{i,j}$ der Kosten, das Objekt i der Messung j zugewiesen wird und sei $l_{i,0}$ der Kosten für den Fall, dass i nicht detektiert wurde. Man bilde also die Kostenmatrix $\mathbf{L} \in \mathbb{R}^{n_k \times (m_k + n_k)}$.

$$\mathbf{L} = \begin{bmatrix} l_{1,1} & l_{1,2} & l_{1,3} & l_{1,0} & \infty \\ l_{2,1} & l_{2,2} & l_{2,3} & \infty & l_{2,0} \end{bmatrix}$$

Die Koordinaten eines Objekts können entweder genau einer Messung entsprechen, oder nicht detektiert werden. Eine Messung kann maximal einem Objekt zugewiesen werden. Die " ∞ " Einträge der Matrix schließen sinnlose Assoziationen aus. Für die Lösung des Problems, suggeriert (Mills-Tettey, Stentz, & Dias, 2007) die Anwendung eines *Hungarian-Algorithmus*, der das folgende Optimierungsproblem erfasst:

$$\begin{aligned} & \underset{\mathbf{A}}{\text{minimiere}} \quad \text{tr}(\mathbf{A}^T \mathbf{L}) \\ & \text{u.d.v} \quad \sum_j \mathbf{A}_{i,j} = 1, \quad \sum_i \mathbf{A}_{i,j} = 1. \end{aligned} \quad (5)$$

Matrix \mathbf{A} enthält lediglich die Werte 0 und 1 und die Summe ihrer Spalten und Zeilen ergeben immer genau 1. Die Anweisung $\text{tr}(\cdot)$ repräsentiert die Spur einer Matrix und daher die Operation $\text{tr}(\mathbf{A}^T \mathbf{L})$ ergibt die Assoziationskosten einer bestimmten Zuordnung.

Zur Bestimmung der Kostenmatrix Elementen $l_{i,j}$ stellt (of Technology, 2019), unter der Voraussetzung eines linearen und Gauß'schen Modells und einer konstanten P^D , die untere Gleichungen vor.

$$l_{i,0} = -\ln(1 - P^D) \quad (6)$$

$$l_{i,j} = -\left[\ln\left(\frac{P^D}{\lambda_c}\right) - 0.5 \log(\det(2\pi \mathbf{S}_i)) - 0.5(\mathbf{z}_j - \tilde{\mathbf{z}}_i) \mathbf{S}_i^{-1} (\mathbf{z}_j - \tilde{\mathbf{z}}_i) \right],$$

wo die Clutter-Intensität $\lambda_c = 1 - \frac{n_k}{m_k}$, die Koordinaten des Objektes i $\tilde{\mathbf{z}}_i =$

$\mathbf{H} \mathbf{x}_{k|k-1}$ und die Kovarianzinovationsmatrix $\mathbf{S}_i = \mathbf{H} \mathbf{P}_{k|k-1}^i \mathbf{H}^T + \mathbf{R}$ die restlichen Variablen darstellen.

Die Elemente $l_{i,0}$ bilden die Gewichte ab, für den Fall dass i nicht detektiert wurde. Die Elemente $l_{i,j}$ sind, wie bereits erwähnt, die Gewichte, wenn Objekt i der Messung j zugewiesen wird.

Die Lösung dieses Assoziationsproblem wird mithilfe eines Python-Pakets, welches von (Apache Software Foundation, n.d.) zur Verfügung gestellt worden ist.

Zu den Gunsten der Erhaltung einer kompakteren Schreibweise der optimalen

Assoziation, führt man die Variable $\theta \in \mathbb{R}^{n_k}$ ein. $\theta_i = 0$ bedeutet, dass keine neue Detektion für das Objekt i vorhanden ist und $\theta_i = j$ gibt an, dass i der Messung j zugeordnet wurde.

3.4 Kalman-Update

Nach der Ermittlung der wahrscheinlichsten Hypothese, wird eine Korrektur benötigt, um die Fehlervarianz des Schätzers zu reduzieren. Die Korrektur wird mithilfe der Messung der Koordinaten, die wahrscheinlich einem Objekt entspricht, durchgeführt. Folgende Gleichungen repräsentieren diesen Vorgang.

$$\mathbf{K}^i = \mathbf{P}_{k|k-1}^i \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k|k-1}^i \mathbf{H}^T + \mathbf{R})^{-1}. \quad (7)$$

\mathbf{K}^i sei die Kalman-Verstärkung und $\mathbf{P}_{k|k-1}$ die Kovarianzmatrix aus dem Prädiktionsschritt. Letztlich lässt sich den aktuellen Zustand $\underline{x}_{k|k}^i$ beziehungsweise die neue Kovarianzmatrix $\mathbf{P}_{k|k}^i$ folgendermaßen berechnen.

$$\underline{x}_{k|k}^i = \begin{cases} \underline{x}_{k|k-1}^i + \mathbf{K}^i (\underline{z}^{\theta^{*,i}} - \mathbf{H} \underline{x}_{k|k-1}^i) & \text{für } \theta^{*,i} \neq 0 \\ \underline{x}_{k|k-1}^i & \text{für } \theta^{*,i} = 0, \end{cases} \quad (8)$$

$$\mathbf{P}_{k|k}^i = \begin{cases} \mathbf{P}_{k|k-1}^i - \mathbf{K}^i \mathbf{H} \mathbf{P}_{k|k-1}^i & \text{für } \theta^{*,i} \neq 0 \\ \mathbf{P}_{k|k-1}^i & \text{für } \theta^{*,i} = 0. \end{cases}$$

Der Vektor $\underline{z}^{\theta^{*,i}}$ stellt die gemessenen Koordinaten dar, die als wahrscheinlichste Fortsetzung der Bewegung des Objekten i angesehen wird. Ein $\theta^{*,i} \neq 0$ bedeutet, dass das Objekt i detektiert worden ist. Wenn $\theta^{*,i} = 0$ (keine Detektion des Objekts), ist keine Korrektur möglich und daher werden die Werte der Prädiktion weitergereicht.

3.5 M/N Algorithmus

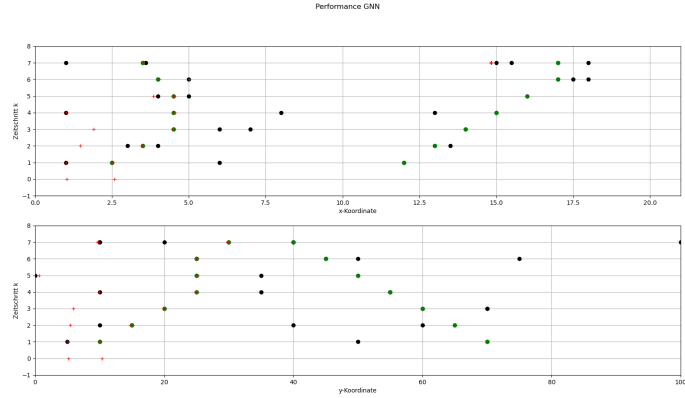
3.6 Startwerte

Die Genauigkeit der Zustandsschätzung des GNN ist von der Wahl der Positionsstartwerten stark abhängig. Bild 1 veranschaulicht diesen Performance Unterschied anhand Testdaten.

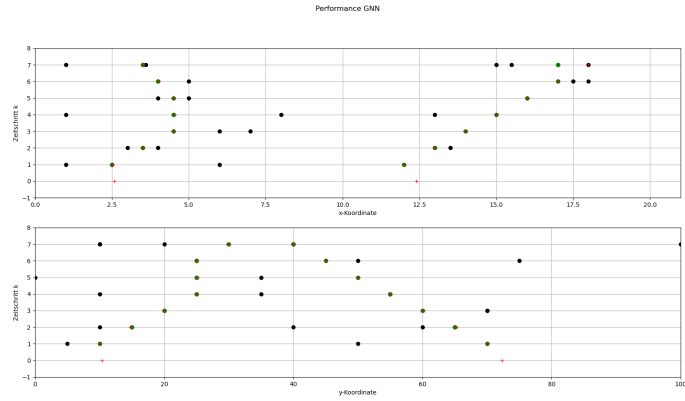
Die Platzierung der Startwerten entscheidet also, ob die Schätzung des GNN als sehr gut oder als sehr schlecht eingestuft werden kann, was den Einsatz des Algorithmus in Frage stellen kann. Aus dem Grund, nutzt man die Zeitverzögerung, die bereits für die Anwendung des *M/N Algorithmus* erforderlich ist, um genaueren Startwerten aus den Messungen zu extrahieren.

Nach der Bestimmung der Objektanzahl, analysiert der *M/N Algorithmus* welche Koordinaten welchem Objektkandidaten gehört und diese werden als x,y Größe für den Zeitschritt $k = N - 1$ verwendet. Werden die richtige Objektkandidaten gefunden, kann man sicherstellen, dass die Startwerte geeignet sind. Falls die gefundenen Kandidaten sich mit den echten unterscheiden, liegen die Startwerte auch falsch und die Qualität der Schätzung des GNN wird je stärker beeinträchtigt je größer diese Abweichung ist.

Die x und y Anfangsgeschwindigkeiten werden immer als null initialisiert.



(a) x,y Schätzung anhand ungeeigneter Startwerten.
Ein von zwei Objekt richtig erkannt



(b) x,y Schätzung anhand geeigneter Startwerten.
Beide Objekte richtig erkannt

Figure 1: GNN Performance, mit geeigneten und ungeeigneten Startwerte und $n_k = \text{const.} = 2$. Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

3.7 Deaths and Births

In einem praktischen Anwendung Szenario bleibt selbstverständlich die Anzahl der Objekte nicht konstant. Daher bei jedem Zeitschritt wird es eine Analyse benötigt, die entscheidet ob ein neues Objekt auf die Bildaufnahme aufgetreten oder verschwunden ist. Man nennt solches Verfahren als *Deaths and Births Analyse*.

Zum Gegenteil des *PHD- Algorithmus* wird dieser Schritt bei dem *GNN* simpel gehalten. Nach der Bestimmung der Objektanzahl n_k , ist ein Objekt *gestorben*, falls $n_k < n_{k-1}$ und wurde *geboren*, falls $n_k > n_{k-1}$.

Für den ersten Fall, vergleicht man die Positionen der Objekte anhand der k und $k - 1$ Zeitschritten. Die Koordinaten aus $k - 1$, die betragsmäßig am weitesten

der neuen Koordinaten k liegen, werden aus dem Zustand gelöscht, daher Objekte, die diesen Koordinaten entsprechen werden als *Tot* angesehen. Für den Fall neuer Geburten, vergleicht man wieder die Beträge der Position über den letzten zwei Zeitschritten. Die Objekte dessen Koordinaten auf $k - 1$ die größten Abstände zum k aufweisen, werden als Objekte, die neulich geboren wurden, erkannt. Deren Positionen werden zusammen mit den Anfangsgeschwindigkeiten in den aktualisierten Zustand eingespeichert.

3.8 GNN Zusammenfassung

Nach der Vorstellung des Aufbaus des *Algorithmus*, ist es möglich seine komplette Struktur darzustellen.

Algorithm 1 GNN Algorithmus

```

1: Initialisieren:  $n_k = -1$ ,  $k = 0$ 
2: while Bilder vorhanden do
3:   Koordinaten vom Zeitschritt  $k$  aus der Detektion erhalten
4:   Messungsanzahl  $m_k$  aus den Koordinaten
5:   if  $k < N$  then
6:     Messungen für M/N Algorithmus laden
7:   if  $k = N$  then
8:      $n_k$  aus dem M/N Algorithmus
9:     Initialisierung  $x_{N|N-1}$ 
10:  if  $k > N$  then
11:    for  $i \leq n_k$  do
12:      Kalman-Prädiktion:  $x_{k|k-1}$  und  $P_{k|k-1}$  mithilfe der Gleichungen
13:      3 und 4 ausrechnen .
14:      Kostenmatrix mithilfe der Gleichung (6) erstellen
15:      Wahrscheinlichste Hypothese mithilfe eines Hungarian-
16:      Algorithmus
17:      Kalman-Update:  $x_{k|k}$  und  $P_{k|k}$  mithilfe der Gleichungen 8 kor-
18:      rigieren
19:       $x_{k|k}$  ausgeben
20:      Daten für M/N Algorithmus aktualisieren
21:       $n_{k+1}$  bestimmen mithilfe des M/N Algorithmus
22:       $x_{k+1|k}$  mithilfe Deaths and Births Analyse ergänzen
23:       $k = k + 1$ 

```

4 Ergebnisauswertung auf Testdaten

Wie bereits auf Bild 1 gezeigt, findet der *GNN* das richtige Tracking anhand der Testdaten lediglich, wenn der Startwert den echten Positionen annähert. Bild 2 zeigt die Schätzung der x,y Koordinaten. Man beobachte, dass in der Regel, die Zustände beider Objekte nur einen geringen Fehler aufweisen. Der Erfolg der Schätzung kann hauptsächlich der Tatsache zugewiesen werden, dass die Clusters sich nicht wie die angenommene Bewegung der Objekten (*Constant Velocity*

Model) bewegen. Außerdem, findet weder ein *Birth* noch ein *Death* statt und letztlich sind die Hyperparameter genau auf diese Problematik angepasst. In einer praktischen Umwelt sind diese Vereinfachungen allerdings meistens nicht erfüllbar. Eine Performanceanalyse hinsichtlich praxisorientierter Datensätze wird im Kapitel ?? zusammen mit der Performance des *PHD-Filter* vorgestellt.

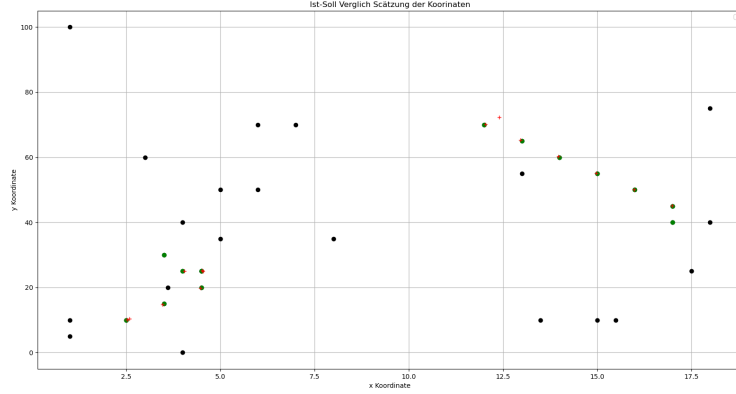


Figure 2: Ist-Soll Vergleich der Koordinatenschätzung: Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

Die erforderliche Hyperparameter, die diese Schätzung erzeugt habe, sind folgendermaßen dargestellt.

- $\mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}$

- $\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

- $\mathbf{P}_0 = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

- $M = 4$

- $N = 5$

- $p^D = 0.99$

- $\eta = 0.08$

Kapitel 1.3.1 beschäftigt sich mit der Variation dieser Parameter und wie sie den Algorithmus beeinflusst.

4.1 Einfluss der Hyperparametern

Bild 3 zeigt wie eine leichte Verringerung der Detektionsrate p^D bereits zu einer falschen Schätzung führen kann. Es liegt daran, dass die $l_{i,0}$ Werte der Kostenmatrix, verglichen mit den $l_{i,j}$ kleiner werden. Infolgedessen wird die Wahrscheinlichkeit einer Nicht-Detektion gegenüber einer Detektion übergewichtet, daher werden die Werte von θ_k^* öfters gleich null sein und die Kalman-Korrektur reicht mehrmals die gleichen Werte der Prädiktion weiter, ohne die Schätzung anhand echter Messungen zu aktualisieren.

Eine Vergrößerung oder Verkleinerung der Werten der Matrizen \mathbf{Q} , \mathbf{R} und

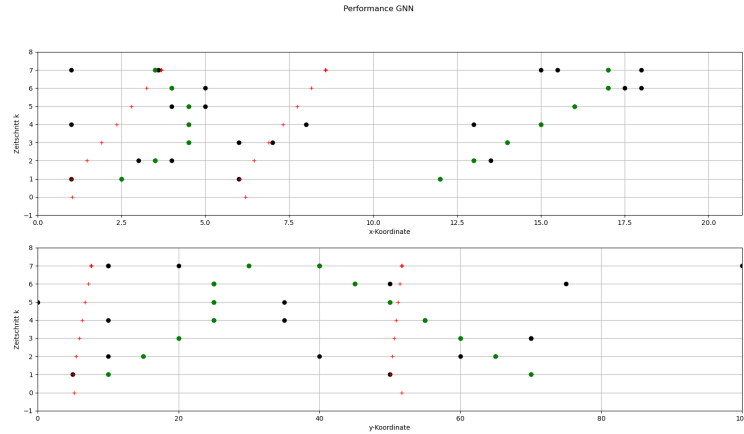


Figure 3: Einfluss $p^D = 0.95$: Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

\mathbf{P}_0 bedeutet eine 1000-Faktor Multiplikation beziehungsweise Division von den gegebenen Matrizen aus dem letzten Kapitel.

Die Varianz des Modellrauschens \mathbf{Q} entscheidet über die Einsatzfähigkeit des Algorithmus. Je mehr Wissen man über die Startwerte und das Systemrauschen verfügt, desto kleiner müssen die Werte von \mathbf{Q} gewählt werden. Eine betragsmäßig höhere \mathbf{Q} erlaubt dem Kalman-Filter eine höhere Abweichung des Zustands über Zeitschritten hinaus, daher werden höhere *Sprünge* ermöglicht, was sich für ein Szenario aneignet, wo weniger Wissen über die Anfangsbedingungen vorhanden ist. Bild 4 veranschaulicht: wenn \mathbf{Q} zu gering gewählt wird, verringert sich erheblich die Reaktionszeit auf unerwartete Messungen des *GNN*.

Die Varianz des Messrauschens \mathbf{R} entscheidet wie stark der Algorithmus von den Messungen abhängig gestaltet wird. Je kleiner die Werte von dieser Matrix, desto geringer ist die Abweichung der Schätzung bezüglich der Messungen. Dies führt zu einer Überanpassungsgefahr, da bei einer Clutter-behateten Umwelt, der unerwünschte Einfluss der Clutters übergewichtet wird. Allerdings für große \mathbf{R} Werte, wird der Einfluss der Messungen derart untergewichtet, dass der Algorithmus Schätzungen mit enormer Abweichung zur Realität suggeriert. Dieser Effekt wird genau auf Bild 5 erkannt, wo der Einfluss der Messungen nach wenigen Zeitschritten schon beinahe vernachlässigt wird.

Ähnlich wie bei der Wahl der Varianz des Modellrauschens, sind die Werte

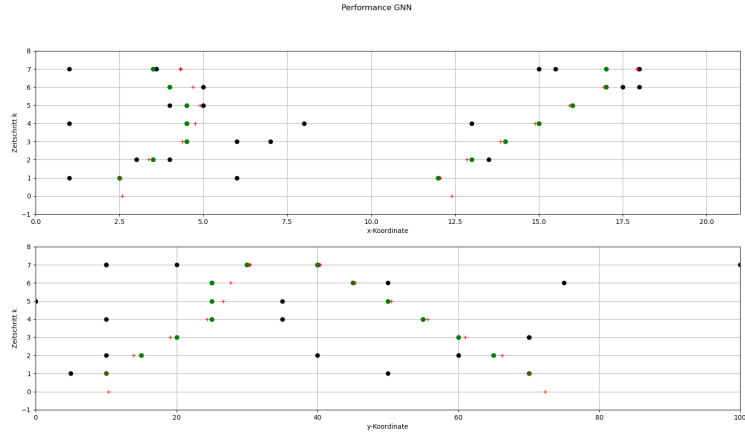


Figure 4: Einfluss einer Verringerung der Werten von \mathbf{Q} : Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

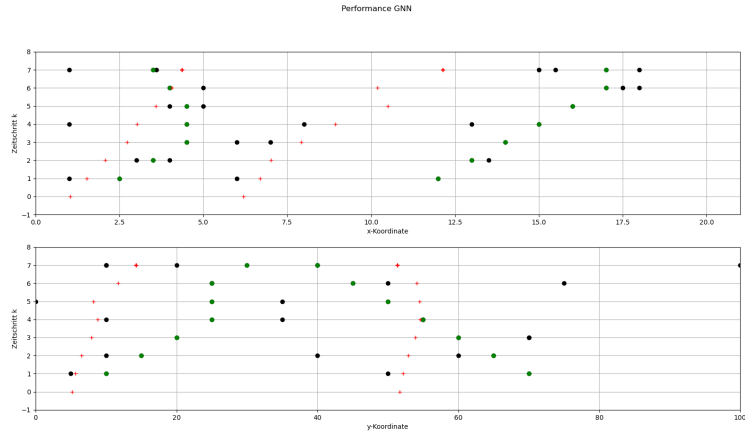
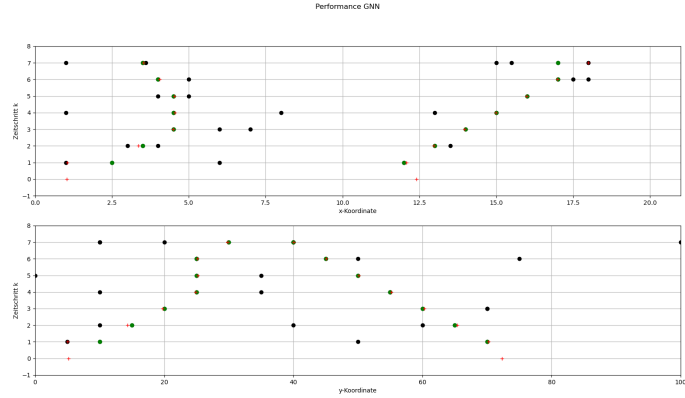
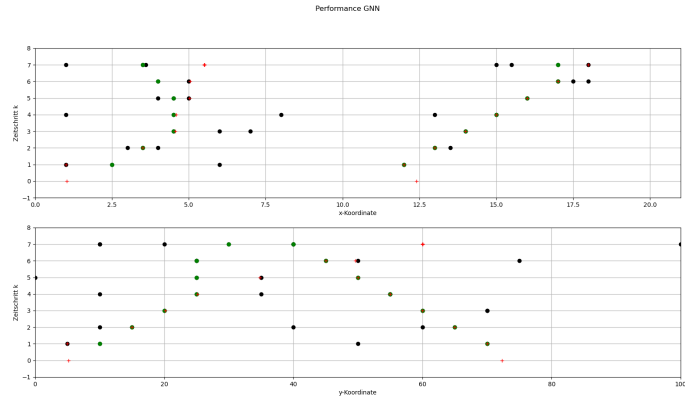


Figure 5: Einfluss einer Vergrößerung der Werten von \mathbf{R} : Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

der Kovarianzmatrix des Schätzfehlers \mathbf{P}_0 abhängig von dem Vorwissen des Anfangsbereichs. Ist kein Vorwissen vorhanden, werden höhere Werte von \mathbf{P} erforderlich, die noch ermöglichen das Tracking der Objekte. Da, man tatsächlich mithilfe der Zeitverzögerung, die für den *M/N Algorithmus* erforderlich ist, Informationen über den Bereich der Anfangsbedingung erhält, empfiehlt (Lenz, 2019) diesen für die Wahl der \mathbf{P}_0 zu nutzen. Bild 6 zeigt die Qualitätsminderung für die Wahl ungeeigneter Anfangswerten von \mathbf{P}_0 .



(a) Einfluss einer Verkleinerung der Werten von \mathbf{P}_0



(b) Einfluss einer Vergrößerung der Weten von \mathbf{P}_0

Figure 6: GNN Performance, mit unterschiedlichen Anfangswerten \mathbf{P}_0 . Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

5 Einschränkung des Algorithmus

Der Entwurf eines rechnengünstigen und schnellen Algorithmus wie der *GNN* wird auf Kosten bestimmter Einschränkungen ausgelegt. Folgende Liste präsentiert ein Fazit über die Haupteinschränkungen und warum für bestimmte marianischen Szenarien die Anwendung des *GNN* problematisch sein kann.

- *GNN Prunning*: Die Berücksichtigung von lediglich der wahrscheinlichsten Datenassoziation ist der Grund warum der *GNN* simpel und schnell ist. Allerdings es reichen wenige Zeitschritten, wo die wahrscheinlichste Hypothese nicht der Bewegung eines Objekts entspricht, um die Verfolgung dieses Objekts komplett zu unterbrechen. Eine maritimische Umwelt, wo viele Clutters vorhanden sind können den Einsatz dieses Algorithmus verhindern. Aus dem Grund, der Einsatz ist nur zu gereicht fertigen, wenn

die Detektionsqualität als hoch eingestuft wird.

- Der Algorithmus geht prinzipiell von einer bekannten und konstante Objektanzahl aus. Damit man dies umgehen kann, integriert man den *M/N Algorithmus* was nur eine grobe Einschätzung des n_k abliefert. Außerdem, wird eine Zeitverzögerung von einigen Frames erforderlich, bevor der Zustand ausgegeben wird.
- *Deaths and Births Model*. Dies ist ebenso ein wichtiges Element, das der *GNN* prinzipiell nicht berücksichtigt. Anhand dieser Arbeit werden die Annahmen im Kapitel 1.2.7 getroffen. Diese Annahmen können ebenfalls die Trackingsqualität vermindern.
- *Hohe Detektionsrate erforderlich*: Aufgrund der Auslegung der Gleichung der Kostenmatrix, wird eine hohe Detektionsrate erforderlich, damit der *Hungarian Algorithmus* die Messung des Objekts nicht als eine Nichtdetektion einstuft. In der praktischen Anwendung bedeutet, dass eine hohe Kameraqualität erwünscht ist.
- *Auswahl der Kalman-Filter Parametern*: Die Wahl der Varianzmatrizen des Kalman-Filters führen zu einer Überanpassungsgefahr an bestimmten Szenarien. Wenn die Umweltbedingungen sich viel unterscheiden von Bedingungen, an den die Parameter festgelegt wurden, werden die Ergebnisse von der Realität abweichen. Bei der Wahl dieser Parameter wird eine bestimmte Erfahrung des Benutzers erforderlich.
- *Constant Velocity Model*: Dies ist eine Einschränkung die eigentlich alle Algorithmen, die von diesem Modell ausgehen, betrifft. Wenn Objekten sich mit einer großen Beschleunigung bewegen, wird deren Verfolgung stark beeinträchtigt. Außerdem, wenn Clutters nicht zufällig vorkommen, sondern sich näherungsweise konstant und linear "bewegen", werden die vermutlich als Objekt erkannt.

References

- Apache Software Foundation. (n.d.). *Hadoop*. Retrieved from <https://hadoop.apache.org>
- Lenz. (2019). Identification dynamischer systemen. In *Identification dynamischer systemen*. TU Darmstadt.
- Mills-Tettey, G. A., Stentz, A., & Dias, M. (2007). The dynamic hungarian algorithm for the assignment problem with changing costs..
- of Technology, C. U. (2019). *Multiple object tracking* [Chanel]. <https://www.youtube.com/channel/UCa2-fpj6AV8T6JK1uTRuFpw>. (In comment section, accessed on 2018/12/04)