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 rechnengünstigste 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 n_k in jedem Zeitschritt 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 z_k mit x und y Koordinaten aus der Objektdetektion. Dies enthält sowohl Clutters wie die Koordinaten der echten Objekten. 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 Wahrscheinlich den Clutters zugewiesen. Wir definieren m_k als die gesamte Anzahl der Koordinaten der echten Objekte und Koordinaten der Clutters.
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 z_0 zwei Alternativen besitzen sich mit den Koordinaten der z_1 zu Assozieren.
Laut (Lennart Svenson, 2019) die Gleichung, die die gesamte Anzahl aller Assoziationen N_A für beliebige Objektanzahl n_k und beliebige Messungen m_k

vorgibt lautet:

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

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, wird dieser Algorithmus als *Greedy-Algorithm* bezeichnet.
Mithilfe des Prunings, kann man die folgenden posteriori Wahrscheinlichkeit der Objektzuständen $\underline{x_k}$ darstellen.

$$p_{k|k}^{GNN}(\underline{x_k}) = p_{k|k}^{\theta^*_{1:k}}(\underline{x_k}), \tag{2}$$

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 $p_{k|k}^{\theta^{*,i}_{1:k}}(\underline{x}_k^i)$ für $i = 1, ..., n_k$.

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

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 dem Update 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 Kapital 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 $\mathbf{Q} \in \Re^{4x4}$, die positiv-definite Matrix der Varianz des Messrauschens $\mathbf{R} \in \Re^{2x2}$ und letztlich die symmetrische Objekt Kovarianzmatrix des Systemrauschens $\mathbf{P}^i \in \Re^{2x2}$. 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 $i = 1, ..., n_k$ getrennt auszuführen. Der Einfluss der Wahl der Matrizen $\mathbf{Q}$ und $\mathbf{R}$ sowie des Startwertes der Matrix $\mathbf{P}^i$ werden im Kapitel 6 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 Messungen der aktuellen Ausgangsgrößen noch nicht vorausgesetzt. Die Prädiktion erfolgt mithilfe folgender Gleichungen.

$$\underline{x}^i_{k|k-1} = \mathbf{F}\underline{x}^i_{k-1|k-1}, \tag{3}$$

$$\mathbf{P}^i_{k|k-1} = \mathbf{F}\mathbf{P}^i_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}. \tag{4}$$

Die Notation $[.]_{k|k-1}$ 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, das 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 $\mathbf{L} \in \Re^{n_k \times (m_k + n_k)}$.

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

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:

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

Matrix $\mathbf{A}$ enthält lediglich die Werte 0 und 1 und die Summe ihrer Spalten und Zeilen ergeben immer genau 1. Die Anweisung $tr(.)$ repräsentiert die Spur einer Matrix und daher die Operation $tr(\mathbf{A}^T\mathbf{L})$ ergibt die Assoziationskosten einer bestimmten Zuordnung.
Zur Bestimmung der Kostenmatrix Elementen $l_{i,j}$ stellt (Lennart Svenson, 2019), unter der Voraussetzung eines linearen und Gauß'schen Modells und einer konstanten P^D, die untere Gleichungen vor.

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

$$l_{i,j} = -[ln(\frac{P^D}{\lambda_c}) - 0,5log(det(2\pi\mathbf{S}_i)) - 0.5(\underline{z}_j - \underline{\tilde{z}}_i)\mathbf{S}_i^{-1}(\underline{z}_j - \underline{\tilde{z}}_i)],$$

wo die Clutter-Intensität $\lambda_c = 1 - \frac{n_k}{m_k}$, die Koordinaten des Objektes i $\underline{\tilde{z}}_i = \mathbf{H}\underline{x}_{k|k-1}$ und die Kovarianzinovationsmatrix $\mathbf{S}_i = \mathbf{H}\mathbf{P}^i_{k|k-1}\mathbf{H}^T + \mathbf{R}$ 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, durchgeführt.

Zu den Gunsten der Erhaltung einer kompakteren Schreibweise der optimalen Assoziation, führt man die Variable $\underline{\theta} \in \Re^{n_k}$ 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.

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

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

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

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

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.

4 M/N-Logik

Die *Track-Initiation* ist ein Vorgang, bei dem nicht verknüpfte Messungen zu einem neuem Track erstellt werden. Dieser Prozess soll für den *GNN-Algorithmus* eine geschätzte Objektanzahl n_k ermitteln und somit die Rechenaufwand für das eigentliche Objekt-Tracking verringern. Die *Track-Initiation* lässt sich mit unterschiedlichen Algorithmen realisieren, wie zum Beispiel die *Hough-Transform-Based-Methode*, die *Track-Score-Based-Logik* und die *M/N-Logik*. Für diese Arbeit wurde die *M/N-Logik* verwendet, da dieser Algorithmus häufig genutzt wird, leichter zu Implementieren und schneller ist, und einen geringeren Rechenaufwand hat. Die vorgestellte Methode wurde von (?, ?) inspiriert.

4.1 Grundlegende Idee der M/N Logik

Wie bereits schon erwähnt handelt es sich bei der *M/N-Logik* um einen einfachen Algorithmus, mit dem sich abschätzen lässt ob ein oder mehrere Objekte innerhalb einer Messung, die Clutter enthält, existieren oder nicht. Man geht davon aus, dass Clutter sich unbeständiger verhält als tatsächliche Körper, was sich in

den Messdaten widerspiegelt. Reale Objekte können sich nur mit einer begrenzten Geschwindigkeit bewegen, während Clutter sich sprunghaft verhält. Die Position, wo sich das Objekt im nächsten Zeitschritt befinden könnte, lässt sich somit in einem Gültigkeitsbereich eingrenzen, was in Abbildung 1 gezeigt wird. Diese unterschiedlichen Beständigkeiten können für die *M/N-Logik* genutzt wer-

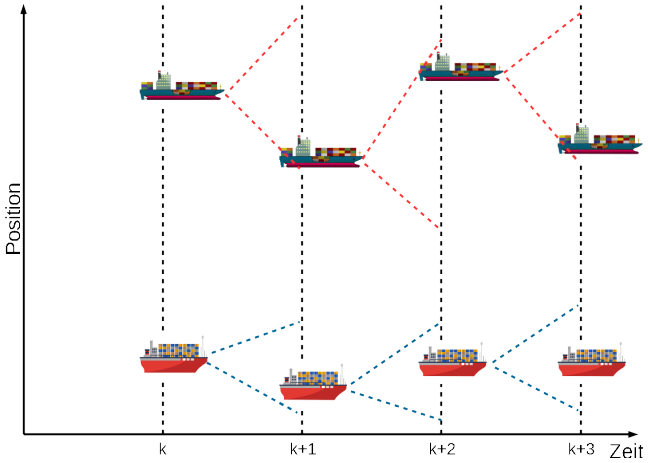


Figure 1: Verhalten des Objekts

den. Weiterhin ist zu erwähnen, dass die Qualität des Tracks bei diesem Algorithmus nur eine untergeordnete Rolle spielt. Es wird nur darauf Wert gelegt, dass die Messpunkte im nächsten Zeitschritt sich im entsprechenden Bereich befinden.
Anhand dieser Ausgangssituation besteht die Aufgabe des Algorithmus darin die Anzahl der erfolgreichen Detektionen M zu zählen, die innerhalb einer festen Anzahl an Scans N vorkommen. Jeder Scan nimmt dabei einen Zeitschritt in Anspruch. Als erfolgreich wird eine Detektion eingestuft, wenn die Messpunkte von zwei Zeitschritten miteinander in Verbindung gebracht werden können,d.h. wenn es einen Messpunkt gibt, der sich im nächsten Zeitschritt in einem Gültigkeitsbereich befindet (Abbildung 1). Liegen keine passenden Messpunkte vor, so wird dies als Fehldetektion gewertet. Wird die Mindestanzahl M an erfolgreichen Detektionen erreicht, liegt ein Objekt vor. Werden am Ende der N Scans nicht ausreichend viele Detektionen erreicht, handelt es sich wahrscheinlich nicht um ein Objekt, sondern um Messpunkte die durch Clutter verursacht wurden.
Zu diesem Zweck wird der Status eines Tracks (bestätigt, vorläufig, vernachlässigt) eingeführt. Ein Track gilt als bestätigt, wenn mindestens M mal innerhalb von N Scans das Objekt detektiert wurde. Wenn keine Möglichkeit besteht diese Häufigkeit zu erreichen, kann der Track vernachlässigt werden. Messpunkte, die der Algorithmus noch verarbeitet, sind als vorläufige einzustufen, weil sie nicht die Bedingungen erfüllen um als bestätigt oder vernachlässigt zu gelten. Abbildung 2 veranschaulicht welchen Status ein Track während des Algorithmus annehmen kann am Beispiel einer *3/4-Logik* (3 benötigte Detektionen aus 4 Scans).

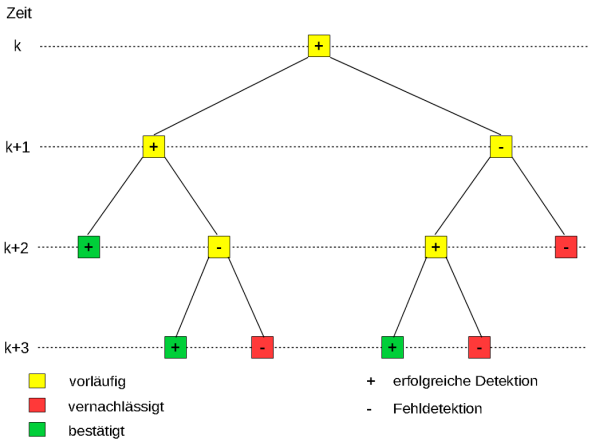


Figure 2: Status des Tracks

4.2 Schwächen der M/N-Logik

Zwei Szenarien muss man immer beachten, wenn die *M/N-Logik* genutzt wird. Erstens kann es bei der Messung dazu kommen, dass nicht nur sehr viel Clutter auftritt, sondern diese während der Messzeit so angeordnet sind wie Objekte, d.h. der Clutter ist sehr beständig. In diesem Fall wird die Anzahl der Objekte überschätzt.
Das zweite Szenario verursacht eine Unterschätzung der Objektanzahl. Dies wird hervorgerufen, wenn sich die Messdaten der Zielkörper unbeständig verhalten. Mögliche Gründe dafür sind häufig vorkommende Fehldetektion, tatsächliche Objekte die sich sehr schnell bewegen und gleichzeitig ihre Geschwindigkeit ändern. Letzteres spielt für diese Arbeit ein untergeordnete Rolle.

4.3 Umsetzung

Dieses Unterkapitel befasst sich mit der Umsetzung des Algorithmus für das Projekt. Im folgenden werden die Wahl von M und N, eine grobe Implementierung und die Festlegung Gültigkeitsbereichs, um Objektmesspunkte von Clutter zu unterscheiden, vorgestellt.

4.3.1 Wahl von M und N

Bei der Entscheidung für ein passendes M/N-Verhältnis sind Faktoren wie Realisierbarkeit, Qualität des Ergebnis und Rechenaufwand bzw. Rechendauer von großer Bedeutung. Hierfür ist eine Eingrenzung von $1/2 < M/N < 1$ sinnvoll, dabei sind M und N natürliche Zahlen. Die Anzahl an erfolgreichen Detektionen M muss auf jeden Fall kleiner sein als die Scan-Häufigkeit. Der Grund hierfür ist, dass eine hundertprozentige Detektionsrate unrealistisch ist und der Algorithmus zu viele Messpunkte als Clutter einstufen würde. Den gegenteiligen Effekt, Clutter wird als Objekt behandelt, würde man erhalten, wenn M/N zu niedrig ist. Mit einer *2/3-Logik* erhält man einen Algorithmus mit minimaler Rechenzeit für eine schnelle Abschätzung, allerdings auf Kosten der Qualität. Mit größer werdendem M/N-Verhältnis wird das Ergebnis genauer,

jedoch stellt man erfahrungsgemäß fest, dass ab $N = 5$ (*4/5-Logik*) keine signifikante Verbesserungen auftreten, die eine weitere Erhöhung der Scan-Dauer bzw.-Anzahl rechtfertigen. Demzufolge ist die *3/4-Logik* eine weiter Option mit einem qualitativ besseren Ergebnis. Für das Projektseminar hat man sich für $M = 3$ und $N = 4$ entschieden, weil bei einer maritimen Hinderniserfassung eine sehr schnelle Objektabschätzung nicht zwingend nötig ist und man mehr Wert auf die Qualität gelegt hat.

4.3.2 Implementierung

Die grundlegend Struktur der *M/N-Logik* kann wie folgt implementiert werden.

Algorithm 1 M/N-Algorithmus

```
Initialisieren:
n_k = 0 (Anzahl der Objekte)
m_k = 1 (Detektionen)
mbar = 0 (Fehldetektionen)
Entscheidung zum Zeitpunkt k
for alle vorläufigen Tracks do
    if Messpunkt im Gültigkeitsereich then
        m_k ← m_k + 1
    else (Keine Messpunkte im Gültigkeitsereich)
        mbar ← mbar + 1
    if m_k ≥ M then (d.h. ausreichend Detektionen)
        Bestätige Track als Objekt
        n_k ← n_k + 1
    else if mbar ≥ N − M then (d.h. zu viele Fehldetektionen)
        Vernachlässige Track
    else
        Fortfahren
```

Für die Startwerte des *GNN* wurde der Algorithmus etwas modifiziert. Anstatt vom aktuellen Zeitpunkt ausgehend einen passenden Messpunkt im nächsten Zeitschritt zu finden, wie es in Abbildung 1 der Fall ist, soll die *M/N-Logik* zum Zeitpunkt k in den vergangenen Messwerten die passenden Daten ermitteln. Abbildung 3 zeigt das prinzipielle Vorgehen der rückwärts laufenden Suche. Der Grund für diese Modifikation ist, dass dadurch die Ermittlung der Startwerte für den *GNN-Algorithmus* vereinfacht wird. Man übernimmt die Koordinaten eines Tracks zum aktuellen Zeitpunkt k, das als Objekt bestätigt wurde und übergibt sie dem *GNN*. Bei einer vorwärtslaufenden *M/N-Logik* besteht die Möglichkeit, dass man am Ende mehrere Kandidaten pro Track bzw. Objekt hat und somit ein weiterer Algorithmus für die Auswahl der Anfangswerte nötig wäre.

4.3.3 Gültigkeitsbereich

Der Gültigkeitsbereich wird genutzt, um darüber zu entscheiden ob eine erfolgreiche oder fehlgeschlagene Detektion vorliegt. Unter der Annahme, dass die Geschwindigkeiten bekannt sind, gilt der Zusammenhang

$$x_{k-1} = x_k - Tv_{k-1}^x \tag{9}$$

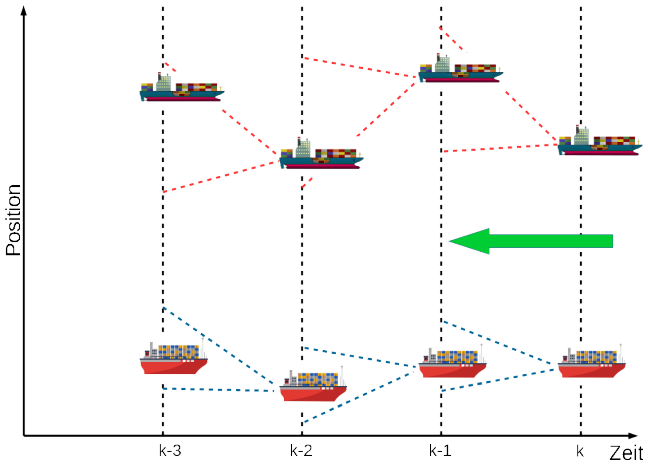


Figure 3: Rückwärtslaufende Suchrichtung der *M/N-Logik*

$$y_{k-1} = y_k - Tv_{k-1}^y.$$

Allerdings sind diese während der ersten N Zeitschritte, dem sogenannten Warmlaufen, nicht gegeben, weshalb Anfangs nicht auf die Geschwindigkeiten zurück gegriffen werden kann um einen Gültigkeitsbereich zu bestimmen. Stattdessen musste über die Koordinaten der Messdaten eine Schwelle für diesen Bereich ermittelt werden. Als Bedingung für die Gültigkeit wurde eine Umgebung u mit

$$\begin{aligned} |z_k^x - z_{k-1}^x| &\leq u \\ |z_k^y - z_{k-1}^y| &\leq u \end{aligned} \tag{10}$$

festgelegt, die den maximalen Abstand zweier Messpunkte angibt. z^x bzw. z^y beschreibt hierbei die x- bzw. y-Koordinate eines Messpunktes. Ist der Abstand zwischen den Messpunkten ausreichend klein (z_{k-1} liegt also im Gültigkeitsbereich), kann dies als eine erfolgreiche Detektion gewertet werden. Welcher Wert für u genommen wird, hängt von der Anwendung selbst ab und musste durch Trail-and-Error herausgefunden werden. Dazu wurden die Messdaten, bezogen auf die maximal und minimal Wert der Koordinaten, durch

$$\begin{aligned} z_{norm}^x &= \frac{z^x - x_{min}}{x_{max} - x_{min}} \\ z_{norm}^y &= \frac{z^y - y_{min}}{y_{max} - y_{min}} \end{aligned} \tag{11}$$

normiert, um alle Daten auf die gleiche Größenordnung zu bringen. Dadurch wird das Ermitteln eines passenden u deutlich erleichtert. Nach dem der *M/N-Algorithmus* die Messdaten verarbeitet hat, erfolgte eine Rücktransformation mit

$$\begin{aligned} z^x &= (x_{max} - x_{min}) z_{norm}^x + x_{min} \\ z^y &= (y_{max} - y_{min}) z_{norm}^y + y_{min}. \end{aligned} \tag{12}$$

Nach den ersten N Zeitschritten kann der *GNN-Algorithmus* die Geschwindigkeiten der Objekte schätzen. Diese Schätzungen wurden für den Rest der Simulation genutzt um einen genaueren Gültigkeitsbereich zu bestimmen. Mit dem Zusammenhang

$$\begin{aligned} \hat{x}_{k-1} &= z_k^x - T\hat{v}_{k-1}^x \\ \hat{y}_{k-1} &= z_k^y - T\hat{v}_{k-1}^y \end{aligned} \tag{13}$$

kann geschätzt werden, wo sich mögliche Messpunkte befinden. Als Schwelle für den Gültigkeitsbereich wurde der Fehlertoleranz ϵ mit

$$\begin{aligned} |z_{k-1}^x - \hat{x}_{k-1}| &\leq \epsilon \\ |z_{k-1}^y - \hat{y}_{k-1}| &\leq \epsilon \end{aligned} \tag{14}$$

definiert. Ist der Fehler ausreichend klein, dann liegt z_{k-1} im Gültigkeitsbereich und man kann von einer erfolgreichen Detektion aus gehen. Für die Bestimmung von ϵ musste man auch hier nach Trail-and-Error vorgehen. Die Normierung wurden ebenfalls wie oben durchgeführt.

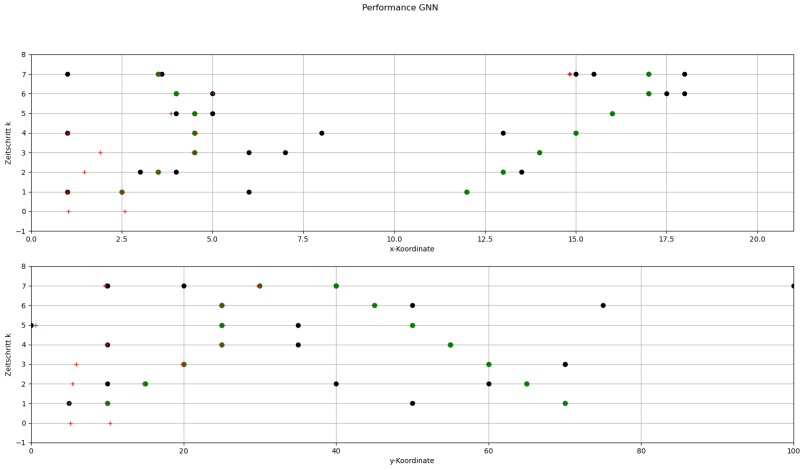
4.4 Startwerte

Die Genauigkeit der Zustandsschätzung des GNN ist von der Wahl der Positionsstartwerten stark abhängig. Bild 4 veranschaulicht diesen Performance Unterschied anhand Testdaten.

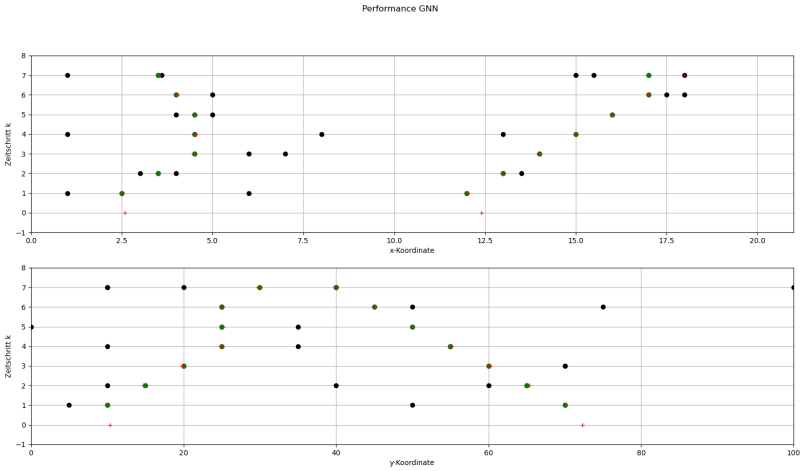
Die Platzierung der Startwerten entscheidet also, ob die Schätzung des GNN als sehr gut oder als sehr schlecht eingestuft werden kann, was den Einsatz des Algorithmus in Frage stellen kann. Aus dem Grund, nutzt man die Zeitverzögerung, die bereits für die Anwendung des *M/N Algorithmus* erforderlich 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 echte unterscheiden, liegen die Startwerte auch falsch und die Qualität der Schätzung des GNN wird je stärker beeinträchtigt je größer diese Abweichung ist.
Die Anfangsgeschwindigkeiten v^x und v^y werden immer als null initialisiert.

4.5 Deaths and Births

In einem praktischen Anwendungsszenario bleibt selbstverständlich die Anzahl der Objekte nicht konstant. Daher in 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*,



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

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. Die Kovarianzmatrix des Schätzfehlers, die dem gelöschten Zustand gehört wir

ebenfalls nicht weiter berücksichtigt.
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. Die Kovarianzmatrix des Schätzfehlers neuer Objekte werden mit $\mathbf{P}_0^i$ initialisiert.

5 *GNN* Zusammenfassung

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

Algorithm 2 GNN Algorithmus - Pseudocode

Initialisieren: $n_k = -1$, $k = 0$
while Bilder vorhanden **do**
    Koordinaten vom Zeitschritt k aus der Detektion erhalten
    Messungsanzahl m_k aus den Koordinaten
    if $k < N$ **then**
        Messungen für *M/N Algorithmus* laden
    if $k = N$ **then**
        n_k aus dem *M/N Algorithmus*
        Initialisierung $\underline{x}_{N|N-1}$
    if $k > N$ **then**
        for $i \leq n_k$ **do**
            Kalman-Prädiktion: $\underline{x}_{k|k-1}$ und $\mathbf{P}_{k|k-1}$ mithilfe der Gleichungen 3 und 4 ausrechnen .
            Kostenmatrix mithilfe der Gleichung (6) erstellen
            Wahrscheinlichste Hypothese mithilfe eines *Hungarian-Algorithmus*
            Kalman-Update: $underlinex_{k|k}$ und $\mathbf{P}_{k|k}$ mithilfe der Gleichungen 8 korrigieren
            $underlinex_{k|k}$ ausgeben
        Daten für *M/N Algorithmus* aktualisieren
        n_{k+1} bestimmen mithilfe des *M/N Algorithmus*
        $\underline{x}_{k+1|k}$ und $\mathbf{P}_{k+1|k}$ mithilfe *Deaths and Births Analyse* ergänzen
    $k = k + 1$

6 Ergebnisauswertung auf Testdaten

Wie bereits auf Bild 4 gezeigt, findet der *GNN* das richtige Tracking anhand der Testdaten lediglich, wenn der Startwert den echten Positionen annähert. Bild 5 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 Clutters 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.

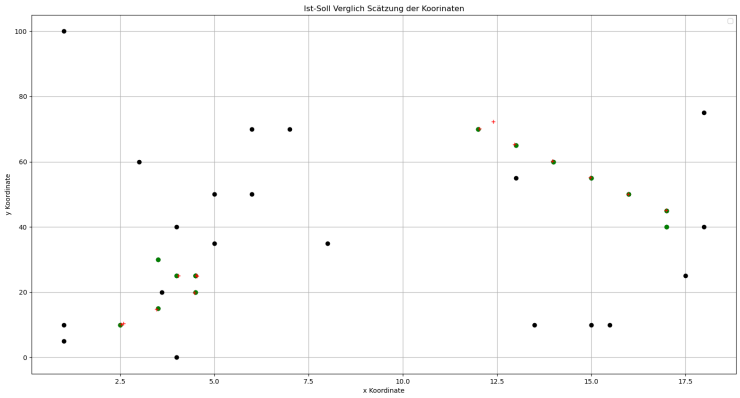


Figure 5: Ist-Soll Vergleich der Koordinatenschätzung: Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

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

- $\mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}$
- $\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- $\mathbf{P}_0 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- $M = 4$
- $N = 5$
- $p^D = 0.99$

- $\eta = 0.08$

Kapitel 6.1 beschäftigt sich mit der Variation dieser Parameter und wie sie den Algorithmus beeinflusst.

6.1 Einfluss der Hyperparametern

Bild 6 zeigt wie eine leichte Verringerung der Detektionsrate p^D bereits zu einer falschen Schätzung führen kann. Es liegt daran, der 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 **Q**, **R** und

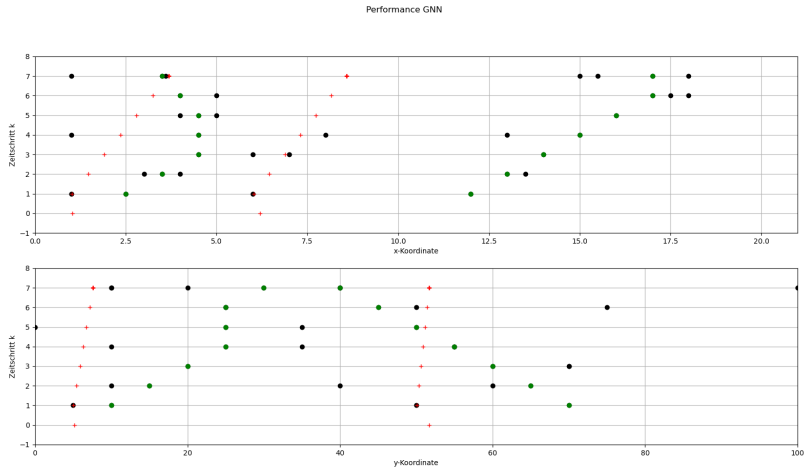


Figure 6: Einfluss $p^D = 0.95$: Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

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

Die Varianz des Modellrauschen **Q** entscheidet über die Einsatzfähigkeit des Algorithmus. Je mehr Wissen man über die Startwerte und das Systemrauschen verfügt, desto kleiner müssen die Werte von **Q** gewählt werden. Eine betragsmäßig höhere **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 7 veranschaulicht: wenn **Q** zu gering gewählt wird, verringert sich erheblich die Reaktionszeit auf unerwartete Messungen des *GNN*.

Die Varianz des Messrauschens **R** entscheidet wie stark der Algorithmus von

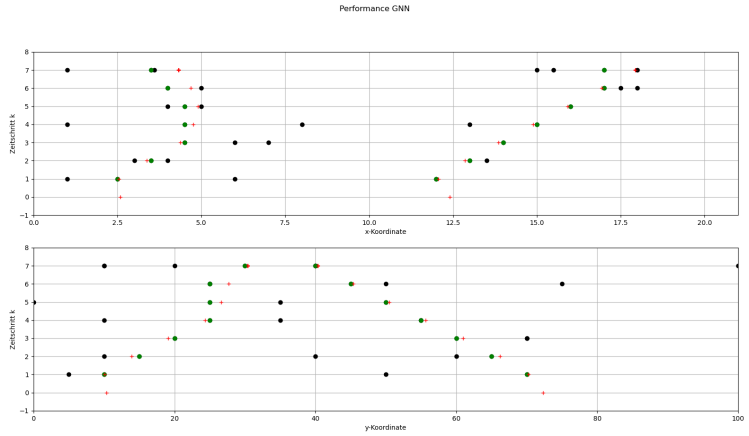


Figure 7: Einfluss einer Verringerung der Werten von **Q**: Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

den Messungen abhängig gestaltet wird. Je kleiner die Werte von dieser Matrix, desto geringerer 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 **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 8 erkannt, wo der Einfluss der Messungen nach wenigen Zeitschritten schon beinah vernachlässigt wird.

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

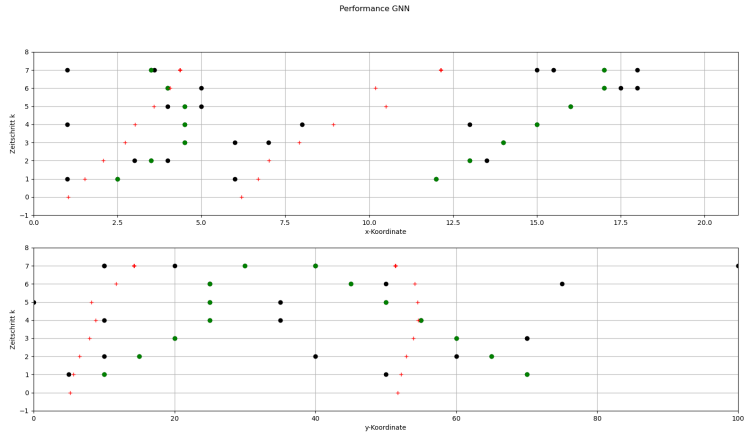
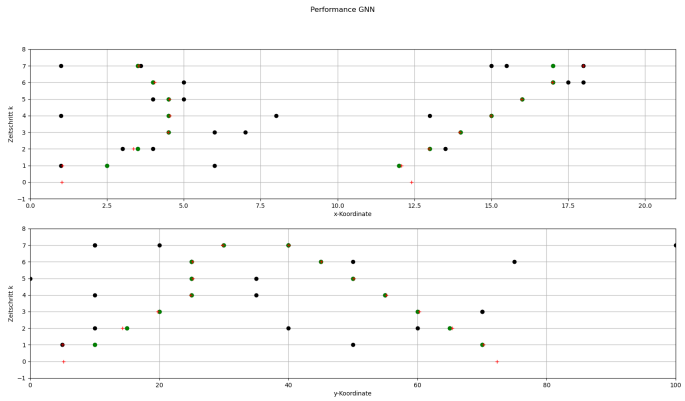
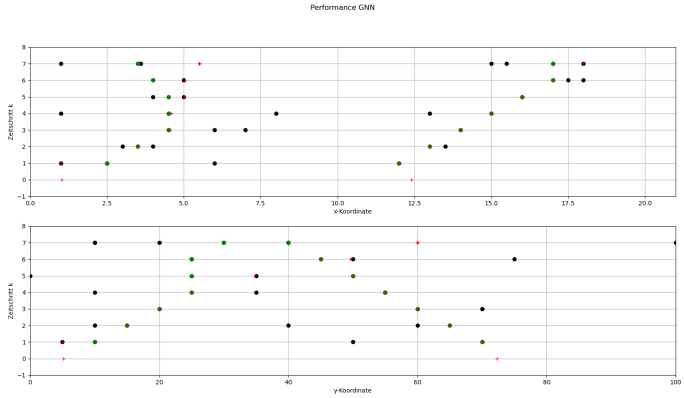


Figure 8: Einfluss einer Vergrößerung der Werten von **R** : Die schwarze Punkte repräsentiert die Clutters, die grüne die echte Positionen der Objekten und die rote Kreuze stellen die geschätzten Positionen des GNN dar

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 das Tracking der Objekte ermöglichen. 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 9 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 9: 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

7 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 maritmischen Szenarien die Anwendung des *GNN* problematisch sein kann.

- *GNN Prunning:* Die Berücksichtigung von lediglich der wahrscheinlichsten Datenassoziation ist der Grund warum der *GNN* simpel uns 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 maritmische Umwelt, wo viele Clutters vorhanden sind können den Einsatz dieses Algorithmus verhindern. Aus dem Grund, der Einsatz ist nur zu gerecht 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 4.5 getroffen. Diese Annahmen können ebenfalls die Trackingsqualität verringern.

- *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 Objekte 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`

Lennart Svenson, C. U. o. T., Karl Ganström. (2019). *Multiple object tracking* [Chanel]. `https://www.youtube.com/channel/UCa2-fpj6AV8T6JK1uTRuFpw`.

Lenz. (2019). Identification dymanischer systemen. In *Identification dymanischer systemen.* TU Darmstadt.

Mills-Tettey, G. A., Stentz, A., & Dias, M. (2007). The dynamic hungarian algorithm for the assignment problem with changing costs..