Sachbericht

Teil 1: Kurzbericht

ZE: Fraunhofer-Institut für offene Kommunikationssysteme (FOKUS)	**Förderkennzeichen:** 16ES1030
Vorhabenbezeichnung: „KIFLEX: Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren“	
Laufzeit des Vorhabens: 01.09.2019 – 31.08.2023	
Berichtszeitraum: 01.09.2019 – 31.08.2023	

1. Aufgabenstellung und wissenschaftlicher und technische Stand

Das Ziel des Projekts „Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren“ (KIFLEX) war die Entwicklung einer neuartigen hochflexiblen Hardware-Plattform samt zugehörigem Software-Framework für KI-basierte Sensorsignalverarbeitung und –fusion im autonomen Fahrzeug. Als Zielanwendung dafür wurde eine kombinierte Lösung aus Umfelderfassung mit Hilfe von LiDAR und Kamera sowie darauf aufbauenden Modulen zur Echtzeitkalibrierung und zur kartenbasierten Lokalisierung avisiert.

Die verwendeten Algorithmen – sowohl für die Sensorsignalauswertung wie deren frühe Fusionierung – basieren zu großen Teilen auf der Verwendung von etablierten Neuronalen Netzen (NN) die dem Stand der Technik entsprechen. Diese Algorithmen erlauben es, das Umfeld äußerst genau zu erfassen und somit eine zuverlässige Grundlage für eine nachfolgende Entscheidungsfindung zu liefern. Während dies im Bildverarbeitungsbereich bereits etabliert ist, zielte die intendierte Inovation des Projektes im Kern auf die Fusion LIDAR-basierter Punktwolken mit den Kamerabildern. Damit zielte KIFLEX den aktuellen Bedarf nach multi-sensorischer Umfelderfassung, die für VDA-Automatisierungsstufen 4 und 5 benötigt wird.

2. Ablauf des Vorhabens

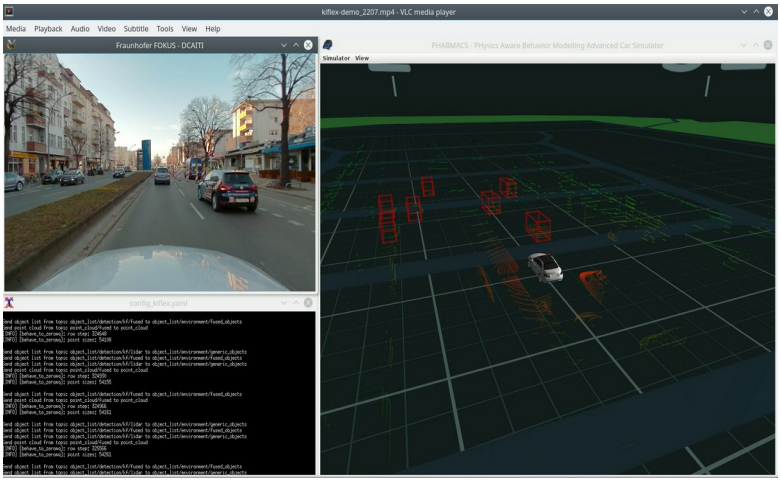
Das Fraunhofer Institut FOKUS war am Förderprojekt KIFLEX mit einer vollen Wissenschaftler-Stelle beteiligt. Unsere im Förderprojekt zu leistenden Inhalte konzentrierten sich auf den Entwurf, die Implementierung und die Validierung der neuronalen Netze zur Perzpetion, die auf dem im Projekt entwickelten Beschleuniger ausgeführt werden, die Entwickung der inovativen Fusion, sowie die Konzeption, Implementierung und Test eines Softwareframeworks zur Sensor-Datenfusion und Umfeldwahrnehmung.

Die verwendeten Algorithmen – sowohl für die Sensorsignalauswertung wie deren frühe Fusionierung – basieren zu großen Teilen auf der Verwendung von Neuronalen Netzen (NN). Diese Algorithmen erlauben es, das Umfeld äußerst genau zu erfassen und somit eine zuverlässige Grundlage für eine nachfolgende Entscheidungsfindung zu liefern.

3. Wesentliche Ergebnisse

Damit die Projektergebnisse große Relevanz für die Praxis besitzen, war es das Ziel eine preisgünstige hochperformante Plattform mit kleiner Leistungsaufnahme in Form eines flexibelen programmierbaren Multi-Core-Deep-Learning-Beschleuniger als ASIC zu entwickeln. Als Kerninovation wurde in KIFLEX ein zukunftsfähiges ASIC entwickelt, das auch nach Integration ins Fahrzeug zum Integrationszeitpunkt noch unbekannte, zukünftige Konzepte und Algorithmen für Neuronale Netze flexibel ausführen kann. Dabei können insbesondere auch die Anpassung an unterschiedliche Betriebszustände des Datenverarbeitungssystems in Abhängigkeit vom Umfeld und Daten berücksichtigt werden.

Das Projekt zielte auf eine ganzheitliche flexible Betrachtung der Datenverarbeitungskette, der darin verwendeten Algorithmen und der ausführenden Hardware. Eine solch enge Verzahnung der Entwicklung aufeinander abgestimmter Algorithmen und Hardware und ist so eine Kerninovation im Bereich des automatisierten Fahrens. Es bietet daher großes Potential zur Steigerung der Effizienz und Leistungsfähigkeit zukünftiger KI-beschleunigender Lösungen. Dies wiederum ermöglicht die effiziente und echtzeitfähige Realisierung neuartiger Algorithmen und bedeutet einen wesentlichen Schritt hin zu deren kommerzieller Nutzbarkeit.



Die Fusion LIDAR-basierter Punktwolken mit den Kamerabildern stellt eine Kerninovation des Projektes dar. KIFLEX bedient damit den aktuellen Bedarf nach multi-sensorischer Umfelderfassung, die für VDA Automatisierungs-stufen 4 und 5 benötigt wird.

Die im Projekt gewonnenen Forschungsergebnisse sollen durch breite Streuung die Forschung entscheidend weiterbringen, und die entwickelte Hardware und Software soll mittelfristig in kommerzielle Produkte der beteiligten Partner münden. Durch die Kombination von Groß- und mittelständischer Industrie und einem KMU (Kleines und Mittleres Unternehmen) sowie Fraunhofer und drei Universitäten demonstrierte KI-FLEX sowohl die Innovationskraft als auch Umsetzungsstärke, um zielstrebig die gesetzten Projektziele zu erreichen.

Schlagwörter: Rekonfigurierbare KI-Hardwareplattform, Sensorfusion, Lidar, Automotive KI-Verfahren, Deep Learning, Umfeldwahrnehmung

Dokumenteninformation

Autoren

Ilja Radusch, Fokus

Bernd Schäufele, Fokus

Förderung

Bundesministerium für Bildung und Forschung

Sachbericht

Teil 2: Eingehende Darstellung

ZE: Fraunhofer-Institut für offene Kommunikationssysteme (FOKUS)	**Förderkennzeichen:** 16ES1030
Vorhabenbezeichnung: „KIFLEX: Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren“	
Laufzeit des Vorhabens:	01.09.2019 – 31.08.2023
Berichtszeitraum:	01.09.2019 – 31.08.2023

Kurzfassung

Das Ziel des Projekts „Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren“ (KIFLEX) war die Entwicklung einer neuartigen hochflexiblen Hardware-Plattform samt zugehörigem Software-Framework für KI-basierte Sensorsignalverarbeitung und –fusion im autonomen Fahrzeug. Als Zielanwendung dafür wurde eine kombinierte Lösung aus Umfelderfassung mit Hilfe von LiDAR und Kamera sowie darauf aufbauenden Modulen zur Echtzeitkalibrierung und zur kartenbasierten Lokalisierung implementiert.

Das Hauptarbeiten des Fraunhofer Institut FOKUS im Förderprojekt KIFLEX konzentrierten sich auf den Entwurf, die Implementierung und die Validierung der neuronalen Netze zur Perzpetion, die auf dem im Projekt entwickelten Beschleuniger ausgeführt werden, sowie die Konzeption, Implementierung und Test eines Softwareframeworks zur Sensor-Datenfusion und Umfeldwahrnehmung.

Die die dazu verwendeten Algorithmen – sowohl für die Sensorsignalauswertung wie deren frühe Fusionierung – basieren KI-Netzen. Diese Algorithmen erlauben es, das Umfeld äußerst genau zu erfassen und somit eine zuverlässige Grundlage für eine nachfolgende Entscheidungsfindung zu liefern. Die Fusion LIDAR-basierter Punktwolken mit den Kamerabildern stellt eine Kerninovation des Projektes dar. KIFLEX bedient damit den aktuellen Bedarf nach multi-sensorischer Umfelderfassung, die für VDA-Automatisierungsstufen 4 und 5 benötigt wird.

Zur praxisorientierten Maximierung der Relevanz der Projektergebnisse, war es das Ziel eine preisgünstige hochperformante Hard- und Softwareplattform mit kleiner Leistungsaufnahme in Form eines flexibelen programmierbaren Multi-Core-Deep-Learning-Beschleuniger als ASIC zu entwickeln. Als Kerninovation wurde in KIFLEX ein zukunftsfähiges ASIC entwickelt, das auch nach Integration ins Fahrzeug zum Integrationszeitpunkt noch unbekannte, zukünftige Konzepte und Algorithmen für Neuronale Netze flexibel ausführen kann. Dabei können insbesondere auch die Anpassung an unterschiedliche Betriebszustände des Datenverarbeitungssystems in Abhängigkeit vom Umfeld und Daten berücksichtigt werden.

Das Projekt zielte auf eine ganzheitliche flexible Betrachtung der Datenverarbeitungskette, der darin verwendeten Algorithmen und der ausführenden Hardware. Eine solch enge Verzahnung der Entwicklung aufeinander abgestimmter Algorithmen und Hardware und ist so eine Kerninovation im Bereich des automatisierten Fahrens. Es bietet daher großes Potential zur Steigerung der Effizienz und Leistungsfähigkeit zukünftiger KI-beschleunigender Lösungen. Dies wiederum ermöglicht die effiziente und echtzeitfähige Realisierung neuartiger Algorithmen und bedeutet einen wesentlichen Schritt hin zu deren kommerzieller Nutzbarkeit.

Die im Projekt gewonnenen Forschungsergebnisse sollen durch breite Streuung die Forschung entscheidend weiterbringen, und die entwickelte Hardware und Software soll mittelfristig in kommerzielle Produkte der beteiligten Partner münden. Durch die Kombination von Groß- und mittelständischer Industrie und einem KMU (Kleines und Mittleres Unternehmen) sowie Fraunhofer und drei Universitäten demonstrierte KI-FLEX sowohl die Innovationskraft als auch Umsetzungsstärke, um zielstrebig die gesetzten Projektziele zu erreichen.

Dokumenteninformation

Autoren

Ilja Radusch, Fokus

Bernd Schäufele, Fokus

Förderung

Bundesministerium für Bildung und Forschung

1. Einleitung

Das Ziel des Projekts „Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren" (KIFLEX) war die Entwicklung einer neuartigen hochflexiblen Hardware-Plattform samt zugehörigem Software-Framework für KI-basierte Sensorsignalverarbeitung und –fusion im autonomen Fahrzeug. Als Zielanwendung dafür wurde eine kombinierte Lösung aus Umfelderfassung mit Hilfe von LiDAR und Kamera sowie darauf aufbauenden Modulen zur Echtzeitkalibrierung und zur kartenbasierten Lokalisierung implementiert.

Das Fraunhofer Institut FOKUS war am Förderprojekt KIFLEX mit einer vollen Wissenschaftler-Stelle beteiligt. Unsere im Förderprojekt zu leistenden Inhalte konzentrierten sich auf den Entwurf, die Implementierung und die Validierung der neuronalen Netze zur Perzpetion, die auf dem im Projekt entwickelten Beschleuniger ausgeführt werden, sowie die Konzeption, Implementierung und Test eines Softwareframeworks zur Sensor-Datenfusion und Umfeldwahrnehmung.

Die verwendeten Algorithmen – sowohl für die Sensorsignalauswertung wie deren frühe Fusionierung – basieren zu großen Teilen auf der Verwendung von Neuronalen Netzen (NN). Diese Algorithmen erlauben es, das Umfeld äußerst genau zu erfassen und somit eine zuverlässige Grundlage für eine nachfolgende Entscheidungsfindung zu liefern. Während dies im Bildverarbeitungsbereich bereits etabliert ist, stellt die Fusion LIDAR-basierter Punktwolken mit den Kamerabildern eine Kerninovation des Projektes dar. KIFLEX bedient damit den aktuellen Bedarf nach multi-sensorischer Umfelderfassung, die für VDA-Automatisierungsstufen 4 und 5 benötigt wird.

Damit die Projektergebnisse große Relevanz für die Praxis besitzen, war es das Ziel eine preisgünstige hochperformante Plattform mit kleiner Leistungsaufnahme in Form eines flexibelen programmierbaren Multi-Core-Deep-Learning-Beschleuniger als ASIC zu entwickeln. Als Kerninovation wurde in KIFLEX ein zukunftsfähiges ASIC entwickelt, das auch nach Integration ins Fahrzeug zum Integrationszeitpunkt noch unbekannte, zukünftige Konzepte und Algorithmen für Neuronale Netze flexibel ausführen kann. Dabei können insbesondere auch die Anpassung an unterschiedliche Betriebszustände des Datenverarbeitungssystems in Abhängigkeit vom Umfeld und Daten berücksichtigt werden.

Das Projekt zielte auf eine ganzheitliche flexible Betrachtung der Datenverarbeitungskette, der darin verwendeten Algorithmen und der ausführenden Hardware. Eine solch enge Verzahnung der Entwicklung aufeinander abgestimmter Algorithmen und Hardware und ist so eine Kerninovation im Bereich des automatisierten Fahrens. Es bietet daher großes Potential zur Steigerung der Effizienz und Leistungsfähigkeit zukünftiger KI-beschleunigender Lösungen. Dies wiederum ermöglicht die effiziente und echtzeitfähige Realisierung neuartiger Algorithmen und bedeutet einen wesentlichen Schritt hin zu deren kommerzieller Nutzbarkeit.

Die im Projekt gewonnenen Forschungsergebnisse sollen durch breite Streuung die Forschung entscheidend weiterbringen, und die entwickelte Hardware und Software soll mittelfristig in kommerzielle Produkte der beteiligten Partner münden. Durch die Kombination von Groß- und mittelständischer Industrie und einem KMU (Kleines und Mittleres Unternehmen) sowie Fraunhofer und drei Universitäten demonstrierte KI-FLEX sowohl die Innovationskraft als auch Umsetzungsstärke, um zielstrebig die gesetzten Projektziele zu erreichen.

2. Wissenschaftlich-technischen Ergebnisse und wesentliche Ergebnisse

Die Hauptarbeiten und Ergebnisse des FOKUS im Projekt liegen im

- AP 1.2 der Konzeption des Software-Frameworks
- AP 2.4 Sensor-Datenfusion-Framework-Entwicklung
- AP 2.5 Algorithmen-Entwicklung
- AP 3.2 Konzeption des Sensor-Datenfusion-Frameworks
- AP 3.3 Gesamt-System-Integration.

Die die durchgeführten Arbeiten und erreichten Ergebnisse in Bezug auf die Arbeitspakete sind im Folgenden aufgeführt:

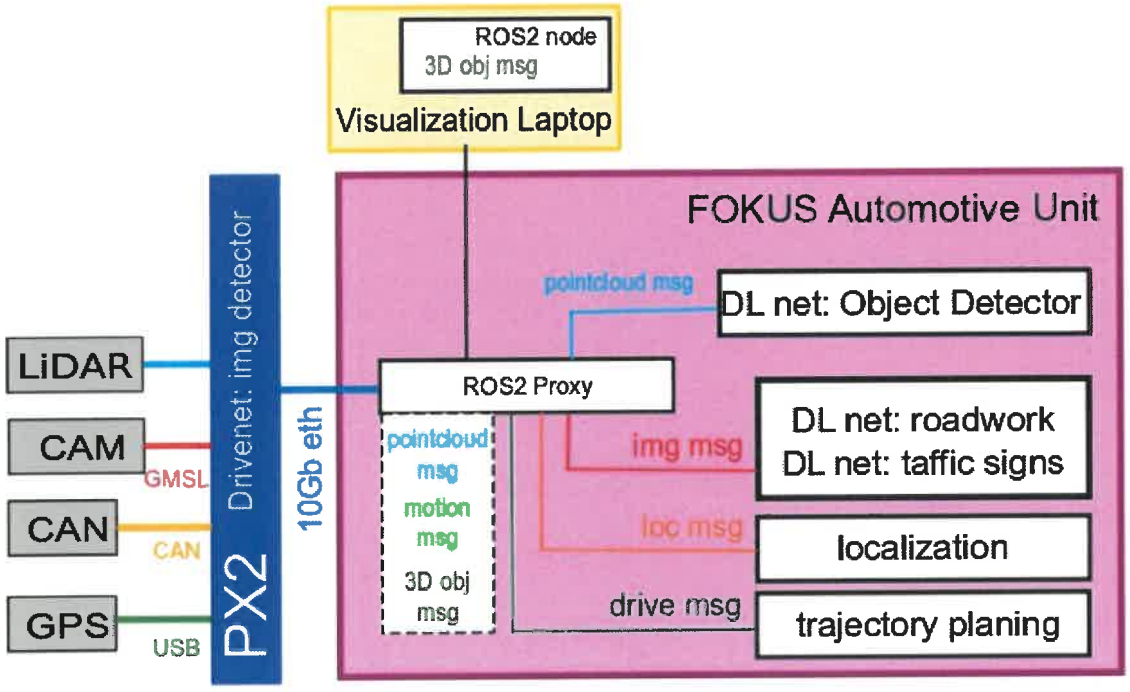
AP 1.2 Konzeption des Software-Frameworks

AP 1.2.2/1.2.4 Konzeption des Software-Frameworks, Spezifikation des Frameworks für Detektion und Klassifikation

Diese APs umfaßten für FOKUS im Projekt die Aufgaben:

- Zuarbeit an Partner IBEO: Anforderungsanalyse der übergreifenden Architektur des Softwareframeworks
- Zuarbeit an Partner IBEO: Anforderungsanalyse der Softwaremodule
- Zuarbeit an Partner IBEO: Anforderungsliste an das Softwareframework bzw. an die Module
- Mitarbeit an der Spezifikation des Softwareframework und der Module

Dazu wurde das FOKUS Behave Framework zur Objektdetektion als mögliche Beispiel-Architektur für das Softwareframework präsentiert und als Zuarbeit in die Analyse und Spezifikation eingebracht.



Das FOKUS Behave System besteht aus:

Sensoreingabeeinheit (PX2)
Aufgabe dieser Einheit ist die Anbindung der Sensoren, Registrieren der Sensordaten und Weiterleitung an die Automotive Unit über 10Gb eth.

- Anbindung LIDAR über 1Gb Eth
- Anbindung Kamera (CAM) über GMSL
- Anbindung CAN-Daten über CAN-Bus
- Anbindung GPS über USB/Serial

Die Sensordaten werden über eine TCP-Socket Verbindung mit 10Gb eth an den ROS2/Socket Proxy der Automotive Unit gesendet. Hier lauft auch ein Image Detektor Netzwerk, dass 2D-Bounding Boxen der Verkehrsobjekte erkennt und über den Proxy an die Automotive Unit weiterleitet.

Die Kommunikation erfolgt über ROS2 messages:

- *pointcloud msg*: 3D LiDAR-Punktwolkendaten (x,y,z,Reflectivity)
- *img msg*: komprimierte Images der Kamera zur Verarbeitung im FlexAISIC img detector net
- *motion msg*: Odometrie-Daten (speed, yaw-rate)
- *loc msg*: Output der Lokalisierung (Position in UTM, heading)
- *3D obj msg*: Informationen über 3D detektierte und klassifizierte Umfeldobjekte
- *drive msg*: Output des Trajectory Planning

FOKUS Automotive Unit
Dies ist ein Automotive-fähiger Linux PC, der eine ROS2 Framework implemetiert und die Daten der Sensoreingabeeinheit sowohl in ROS2-Messages umwandelt und an die weiteren ROS2-Nodes verteilt. Hier laufen weitere Automotive Componenten des FOKUS Versuchsträgers wie die Lokalisierung und Trajektorienplanung. Die wesentlichen auf der Automotive Unit realisierten Module sind

- **ROS2 Proxy:** konvertiert die Sensordaten in ROS2-Messages, die von allen ROS2-Nodes empfangen werden können.
- **ZQM Proxy** zur Anbindung der KIFLEX-Perception Unit
- **DL Network Nodes**: dies sind weitere DL-Netze zur Detektion von dynamische Verkehrsobjekten, Baustellenschildern und Verkehrszeichen.

Visualization Laptop
Zu Zwecken der Visualisierung sowie ggf. zur Aufzeichnung der Daten wird ein Laptop verwendet. Dieser empfängt als ROS2-Node die Sensordaten und Resultate von der Automotive Unit und stellt diese als 3D-Visulalisierung dar.

KI-Flex Softwareframework

Ausgehend von dieser Grundstruktur wurde im Projekt ein Konzept des Softwareframework und seiner erforderlichen Module zusammen mit den Projektpartnern entwickelt und sowie die dazu erforderlichen Anforderungen analysiert. Das entworfene KI-Flex Softwareframework ist im folgenden Bild dargestellt und umfaßt folgende Komponenten:

Kommunikation
Für den Datenaustausch zwischen den Modulen der KI-FLEX-Plattform sind die folgenden Nachrichten-Typen vorgesehen:

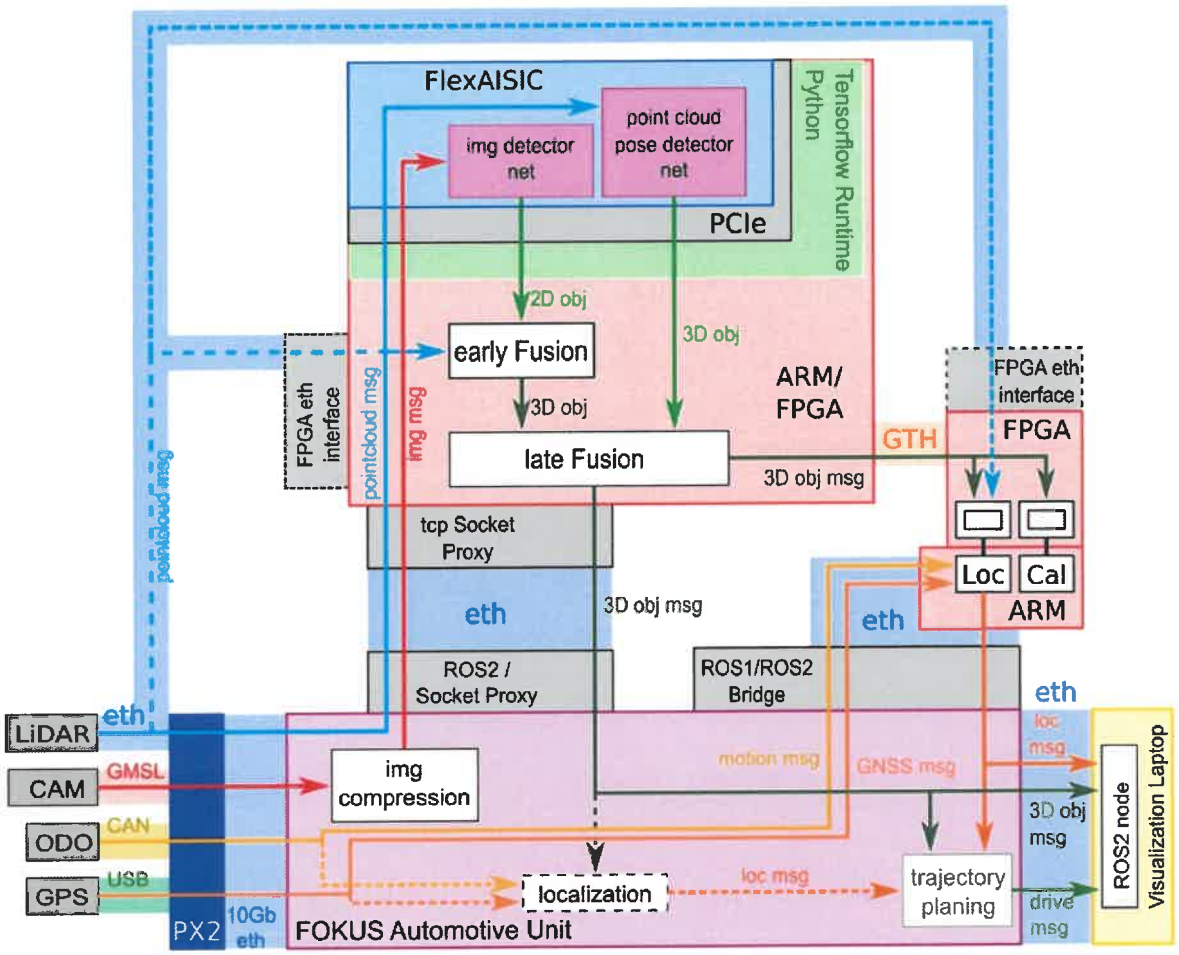
- *pointcloud msg*: 3D LiDAR-Punktwolkendaten (x,y,z,Reflectivity)
- *img msg*: komprimierte Images der Kamera zur Verarbeitung im FlexAISIC img detector net
- *motion msg*: Odometrie-Daten (speed, yaw-rate)
- *loc msg*: Output der Lokalisierung (Position in UTM, heading)
- *3D obj msg*: Informationen über 3D detektierte und klassifizierte Umfeldobjekte
- *drive msg*: Output des Trajectory Planning

Die genaue Zusammensetzung dieser Nachrichtentypen wird im Spezifikationsdokument (E2/E3) spezifiziert werden. Grundsätzlich sind alle dieser Nachrichten mit Zeitstempeln versehen. Die zur Übertragung verwendeten Protokolle variieren je nach Systemkomponente.

Sensoreingabeeinheit (PX2)
Aufgabe dieser Einheit ist die Anbindung der Sensoren, Registrieren der Sensordaten und Weiterleitung an die Automotive Unit über 10Gb eth.

- Anbindung LIDAR über 1Gb Eth
- Anbindung Kamera (CAM) über GMSL
- Anbindung CAN-Daten über CAN-Bus
- Anbindung GPS über USB/Serial

Die Sensordaten werden über eine TCP-Socket Verbindung mit 10Gb eth an den ROS2/Socket Proxy der Automotive Unit gesendet.



FOKUS Automotive Unit

Dies ist ein Automotive-fähiger Linux PC, der die Daten der Sensoreingabeeinheit sowohl in ROS2-Messages umwandelt, als auch über eine TCP-Socket Verbindung in an die Tensorflow Runtime des FlexAISIC zur Verarbeitung in den DL-Netzen weiterleitet und die Resultate der Netze wieder in ROS2-Messages umwandelt und an den Visualization Laptop sendet.
Daneben laufen hier weitere Automotive Componenten des FOKUS Versuchsträgers wie die Lokalisierung und Trajektorienplanung. Die wesentlichen auf der Automotive Unit realisierten Module sind

- **ROS2 Proxy:** konvertiert die Sensordaten in ROS2-Messages, die von allen ROS2-Nodes empfangen werden können.
- **Socket Proxy (ZMQ):** sendet die Sensordaten als ZMQ-Connection an die Tensorflow Runtime des FlexAISIC zur Verarbeitung in den DL-Netzen und empfängt die dort generierten Objektdaten.
- **Img Compression**: komprimiert die Bilder der Kamera , um eine Übertragung über den Socket Proxy an die Tensorflow Runtime des FlexAISIC zur Verarbeitung in den DL-Netzen bei der gegebenen Übertragungsrate zu ermöglichen. Eine Kompression der LiDAR-Daten ist bei einer 1Gb eth-Verbindung nicht notwendig gewesen – kann aber durch Verwendung von Int-Werten auf der Automotive Unit leicht realisiert werden.

Perception Unit: ARM + FPGA + FlexAISIC

Hier läuft die Tensorflow Runtime des FlexAISIC, der in den DL-Netzen die Sensordaten zu 3D Objekten weiterverarbeitet. Die Daten werden von der Automotive Unit über eine Socket Connection empfangen, in den Netzen verarbeitet und in den Fusionskomponenten fusioniert. Der Output wird über die Socket Connection zurück an die Automotive Unit gesendet. Auf die Komponenten der Perception Unit sind die folgenden Aufgaben zu verteilen:

- **TCP Socket Proxy (ZMQ)**: Empfang der Sensordaten und Senden der fusionierten 3D Objekte.
- **Img Decompression**: Dekompression der empfangenen Kamerabilder (zur Verarbeitung in den DL-Netzen des FlexAISIC).
- **Pre- and Postprocessing**: Vor- und Nachverarbeitung der in den neuronalen Netzen verarbeiteten Daten
- **img detector net**: DL-Netz in der Tensorflow Runtime zur Detektion von 2D-Bounding Boxen in den Kamerabildern
- **pointcloud pose detector net**: DL-Netz in der Tensorflow Runtime zur Detektion von 3D-Bounding Boxen mit Heading in den LIDAR-Daten
- **early Fusion**: Fusion der LIDAR-Daten mit den erkannten 2D Bounding Boxen zu 3D Objekten (FOKUS)
- **late Fusion**: Fusion der 3D Objekte aus dem pointcloud pose detector net mit dem Output der early Fusion (Infinion)

Die beiden neuronalen Netze sollen auf dem FlexAISIC in Hardware realisiert werden. Für die restlichen hier aufgezählten Komponenten ist zunächst eine Implementierung auf der ARM-CPU und anschließend, sofern aus Performance-Gründen nötig, eine Auslagerung auf den FPGA vorgesehen.

Visualization Laptop
Zu Zwecken der Visualisierung sowie ggf. zur Aufzeichnung der Daten wird wieder ein Laptop verwendet werden. Dieser empfängt als ROS2-Node die Sensordaten und Resultate von der Automotive Unit und stellt diese als 3D-Visulalisierung dar.

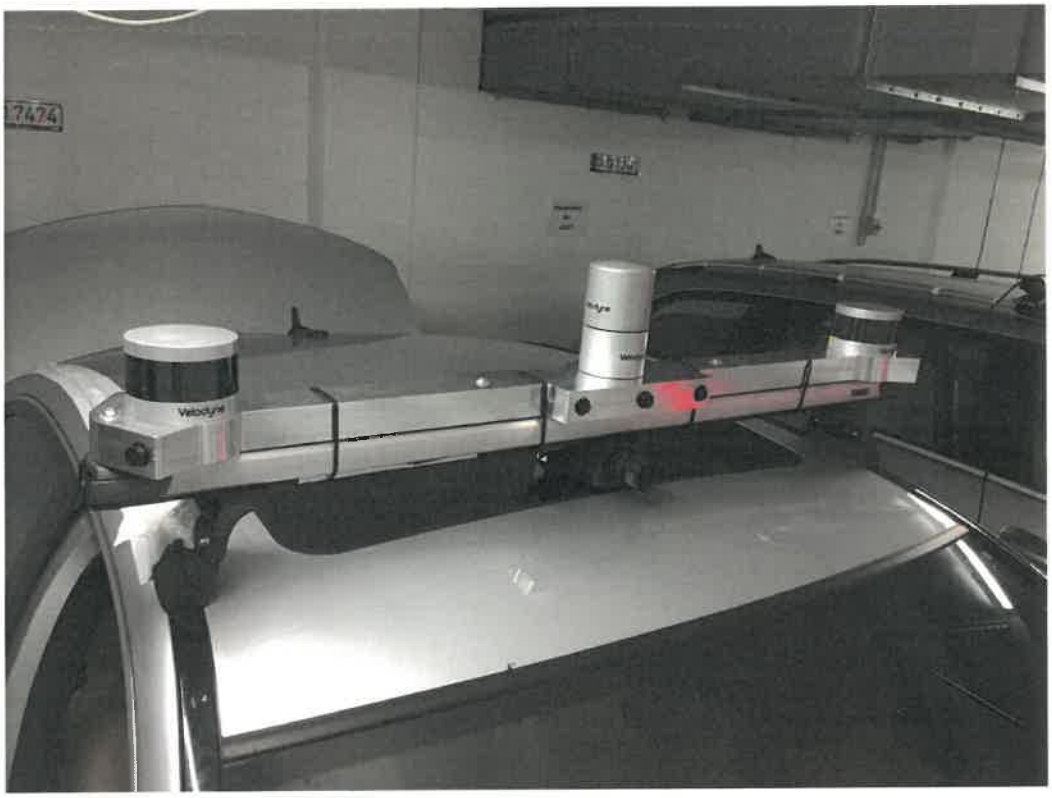
AP 1.2.5. Erkundung möglicher Algorithmen und Ansätze zu Objekterkennung und Datenfusion

Die Aufgabe in diesem AP war die Unterstützung der Auswahl mit Vorwissen Detektionsalgorithmen durch FOKUS.

Durch das Behave-Detektionsframework besitzt FOKUS Vorwissen, daß in die Anforderungsanalyse eingebracht wurde.

Das Behave-System ist einem Versuchsträger implemtiert (s. Abb unten), der mittels Sensoren (Kameras und Lidar) Daten erfasst und Verkehrsobjekte dann mittels DL-Netze detektiert, sowie sich mittels Lidar-Lokalisierung hochgenau verortet.

Die Datenerhebung erfolgt durch ein Sensor-Rig das mit mehreren Kameras und einem 32-Strahl-Lidar ausgestattet ist (s. Abb unten).

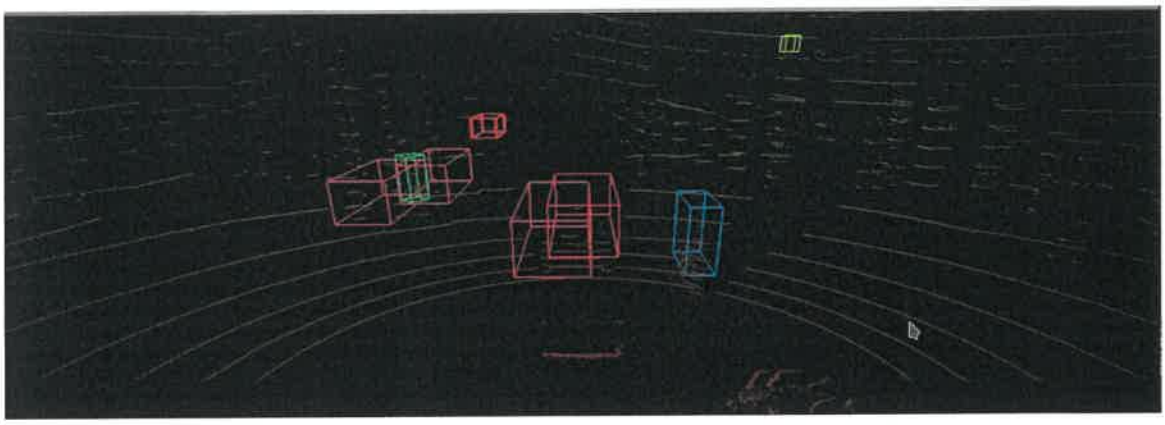


Im System laufen Deep Learning Netze, die auf die Detektion von Verkehrsobjekten in Images trainiert sind (s. Abb).

Eine Netzwerkarchitektur die sich hier bewährt hat und auch als Vorschlag in das Projekt eingebracht wurde, ist das Inception V2 SSD DL-Net. Eine Testversion dieses Netzes wurde den Partnern zur Analyse zur Verfügung gestellt.

Es ist beabsichtigt im Projekt ein weiteres Netz (Lidar-net) zu entwickeln, dass Objekte direkt aus den Lidar-Daten detektiert. Erste Tests haben gezeigt, dass dazu ebenfalls ein Inception V2 SSD DL-Net jedoch mit einem größeren input layer (1000x1000) zu bevorzugen ist. Dieses Netzt wird direkt 3D-Posen (Position, Size, Orientierung) der Objekte erkennen können.

Um von den 2D-Boxen des Image-Net zu 3D-Boxen zu gelangen, ist eine early Fusion notwendig, die die 2D-Boxen und die 3D-Lidar-Punkte zu 3D-Boxen (Position, Size) fusioniert. Dazu wurden im Berichtzeitraum erste Tests durchgeführt und analysiert (s.Abb).



AP 1.2.6 Erkundung möglicher Algorithmen und Ansätze zu Lokalisierung und Echtzeitkalibrierung

In diesem erfolgreich abgeschlossenen AP untersuchte FOKUS ergänzende Vorschläge zu Lokalisierungsalgorithmen als Zuarbeit für den Partner IBEO.

Als Beispiel eines möglichen Lokalisierungsalgorithmus kann die Monte-Carlo-Lokalisierung des FOKUS-Behave-Systems betrachtet werden, das als probabilistischer Schätzer der bedingten Lokalisierungsverteilung in Form eines Partikelfilters ausgelegt ist.

Die Meßgrößen des Schätzers stellen bekannte Landmarken (Poles) einer Karte des Senates des Testgebietes, die aus den 3D-Punktwolken des Lidars segmentierte Landmarkenkandidaten, die relative Fahrzeugbewegung als Odometrie des Versuchsträgers, sowie die zur Initialisierung verwendete GPS-Position dar.

Die zu schätzende Zustandsgröße ist die Egopose des Versuchträgers, die seine globale 2D-Position in UTM-Koordinaten und seine Orientierung als Azimutwinkel in rad (0 rad gleich Ost, positiver mathematischer Drehsinn) beschreibt.

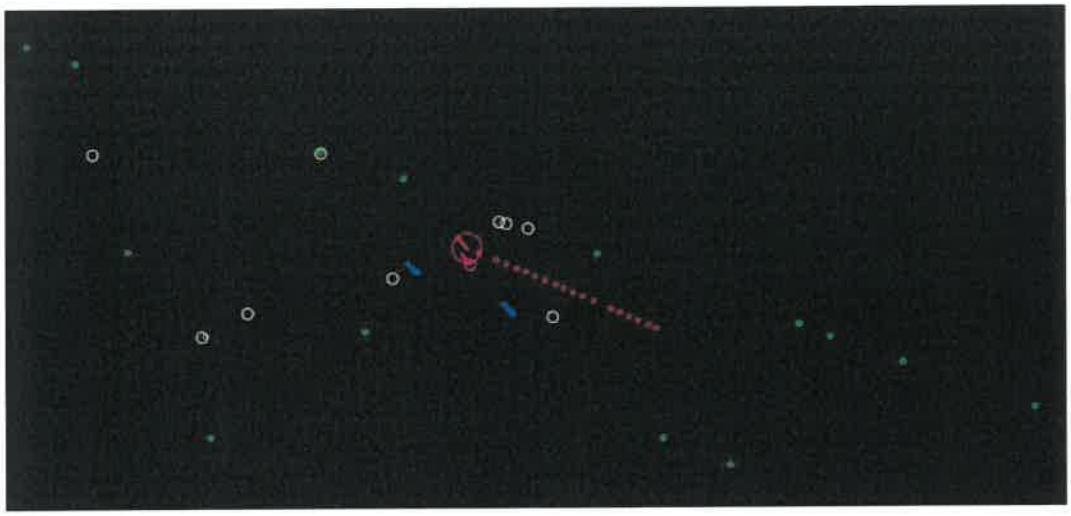
Zur Schätzung verwendet der Algorithmus die Lösung eines Optimierungsproblems, das die gesuchte Pose als Optimum der Minimierung der Abweichung der Positionen der detektierten Landmarken zu den Positionen der assoziierten Landmarken der Karte des Testgebietes.

Der Algorithmus führt die folgenden Arbeitsschritte zur Approximation der bedingten Lokalisierungsverteilung aus:

- **Initialisierung**
  - Die Partikel des Filters, die die stochastischen Hypothesen der Pose des Versuchsträger darstellen, werden mit einer kleinen Streuung um die aktuelle GPS-Position initialisiert.
- **Prediktion**
  - Die Posenhypothesen werden an Hand der Odometriedaten durch Integration der Trajektorie normalverteilt in Position und Orientierung mutiert fortgeschrieben.
- **Messung**
  - ***Erkennung vertikaler Strukturen als Landmarken***
    - Das entwicklete Lokalisierungsverfahren benötigt die Erkennung vertikaler Strukturen, wie beispielsweise Masten von Verkehrszeichen, Lichtsignalanlagen, Beleuchtungsanlagen und Bäume. Zu diesem Zweck wurde ein Algorithmus entwickelt, welcher diese Masten ("Poles") auf Basis von LIDAR-Punktwolken erkennt und der Lokalisierungskomponente zur Verfügung stellt.
    - Diese Erkennungssoftware nutzt die LIDAR-Punktwolke als Eingangsdaten und nimmt zunächst eine Separation des Bodens vor. Anschließend werden die LIDAR-Punkte zu Segmenten zusammengefasst.
    - Für diese Segmente werden diverse Merkmale berechnet, welche zur binären Klassifikation ("Mast", "kein Mast") des Segmentes herangezogen werden. Nach der Klassifikation werden alle Masten über die Middleware an die Lokalisierungskomponente weitergeleitet.
  - Die **Bestimmung der Abweichungen** der detektierten Landmarken von den Kartenlandmarken wird mit Hilfe eines Wahrscheinlichkeitsfeldes der Kartenlandmarken als schneller Matrix-Lookup realisiert, wodurch zur Erhöhung der Echtzeittauglichkeit eine nicht effiziente Nachbarschaftssuche vermieden werden kann. Hypothesen mit geringer Abweichung erhalten dabei höhere Gewichte.

- Zum **Resampling** wird die Partikelpopulation des Filters entsprechend ihrer Gewichtung aus der Messung neu generiert.
- Die **Schätzung** der Ego-Pose erfolgt als Erwartungswert der so aktualisierten Hypothesenverteilung der Partikel des Filters.

Das Verfahren liefert bei genügend detektierbaren Landmarken sehr gute Genauigkeiten, die kleiner als einen halben Meter geschätzt werden können (Eine genaue Angabe ist auf Grund einer fehlenden ground truth nicht möglich). Die untere Abbildung zeigt ein typisches Resultat des Verfahrens (Grün: Kartenlandmarken, weiß: detektierte Landmarken der Lidar-Daten, blau: GPS, pink: integrierte Odometrie, rot: Schätzung des Filters der Position und Orientierung, roter Kreis: Standardabweichung der Partikelverteilung).



Diese hochgenaue neue Selbstlokalisierunglösung verbessert erheblich die bisherige Lokalisierung allein an Hand des GPS, dessen Genauigkeit im Stadtverkehr auf Grund des Multipath-Problems der Satellitennavigation sporadisch große Fehler (kurze Sprünge bis zu 100m) aufweist und damit nicht zur Spurgenauen Lokalisierung und Änderungsdetektion sinnvoll genutzt werden kann.

AP 2.4 Sensor-Datenfusion-Framework-Entwicklung

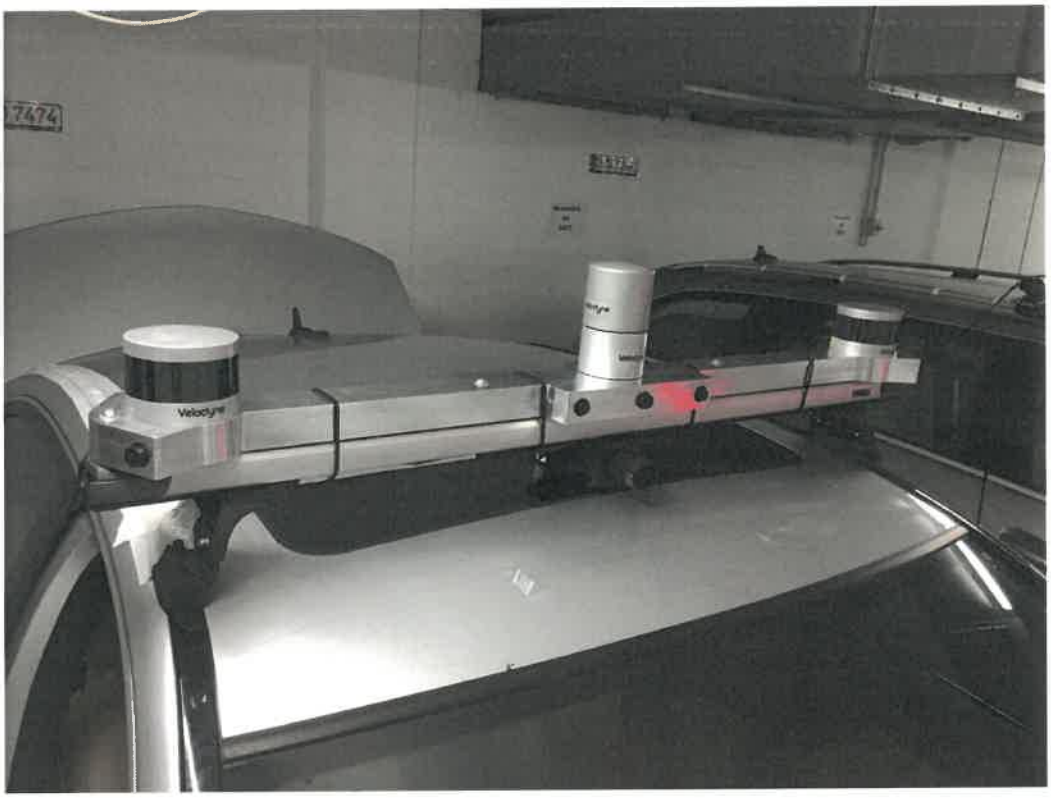
AP 2.4.2 Entwurf des Subsystems zur Umfeldwahrnehmung auf KI-Basis

Die Aufgabe in diesem erfolgreich abgeschlossen AP war die Analyse und der Entwurf eines Softwareframework zur Umfeldwahrnehmung mittels Detektionsalgorithmen.

Durch das Behave-Detektionsframework besitzt FOKUS Vorwissen, daß in die Anforderungsanalyse und den Entwurf der KIFLEX-Detektionsmodule eingebracht wurde.

Das Behave-System ist einem Versuchsträger implementiert (s. Abb unten), der mittels Sensoren (Kameras und Lidar) Daten erfasst und Verkehrsobjekte dann mittels DL-Netze detektiert, sowie sich mittels Lidar-Lokalisierung hochgenau verortet.

Die Datenerhebung erfolgt durch ein Sensor-Rig das mit mehreren Kameras und einem 32-Strahl-Lidar ausgestattet ist (s. Abb unten).



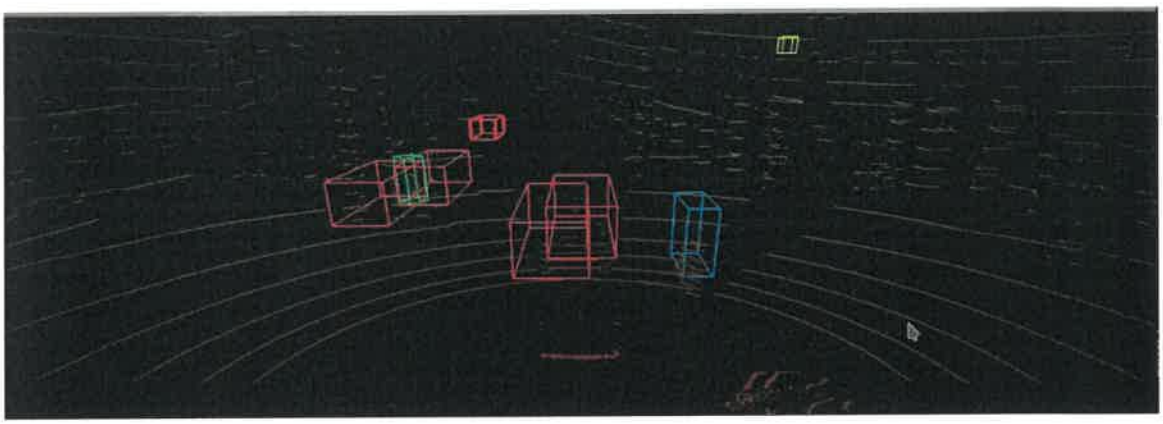
Im System laufen Deep Learning Netze, die auf die Detektion von Verkehrsobjekten in Images trainiert sind (s. Abb).

Eine Netzwerkarchitektur die sich hier bewährt hat und auch als Entwurf in das Projekt eingebracht wurde, ist das Inception V2 SSD DL-Net. Eine Testversion dieses Netzes wurde den Partnern zur Analyse zur Verfügung gestellt.

Der Entwurf sieht außerdem vor, im AP 2.5.1 ein weiteres Netz (Lidar-net) zu entwickeln, dass Objekte direkt aus den Lidar-Daten detektiert. Erste Tests in AP 2.4.2 haben gezeigt, dass dazu ebenfalls ein Inception V2 SSD DL-Net jedoch mit einem größeren input layer (1000x1000) zu

bevorzugen ist. Dieses Netzt wird direkt 3D-Posen (Position, Size, Orientierung) der Objekte erkennen können.

Um von den 2D-Boxen des Image-Net zu 3D-Boxen zu gelangen, ist eine Early Fusion notwendig, die die 2D-Boxen und die 3D-Lidar-Punkte zu 3D-Boxen (Position, Size) fusioniert. Zur Entwurfsentwicklung wurden im Berichtzeitraum erste Tests durchgeführt und analysiert (s.Abb). Dieser Entwurf basiert die Entwicklung des Early Fusion Moduls durch FOKUS im AP 2.4.5 und AP 2.5.2.



AP 2.4.5 Sensor-Datenfusion-Framework-Entwicklung (Fusion)

Im AP entwickelten FOKUS und IFAG ein gemeinsames Softwareframework zur Fusion der Sensordaten und der in der Umfeldwahrnehmung (AP 2.4.2/2.5.1) detektierten Verkehrsobjekte. Das Fusions-Softwareframework unterteilt sich dabei in zwei Phasen der Fusion:

- Early Fusion Modul, Entwicklung FOKUS AP 2.5.2
- Late Fusion Modul, Entwicklung IFAG AP 2.5.3

Das Early Fusion Modul realisiert die frühe Phase der Fusion, in der die 2D-Objektdetektionen der Deep Learning Netze aus AP 2.5.4 in den Kamerabildern mit den 3D-Punktwolken der Lidarsensoren auf Low-Level-Ebene (Rohdaten) fusioniert werden. In Erweiterung dazu, entwickelt IFAG in AP 2.5.3 das Late Fusion Modul, das auf der High-Level-Ebene (Objekte) die verschieden Objektdetektionen fusioniert.

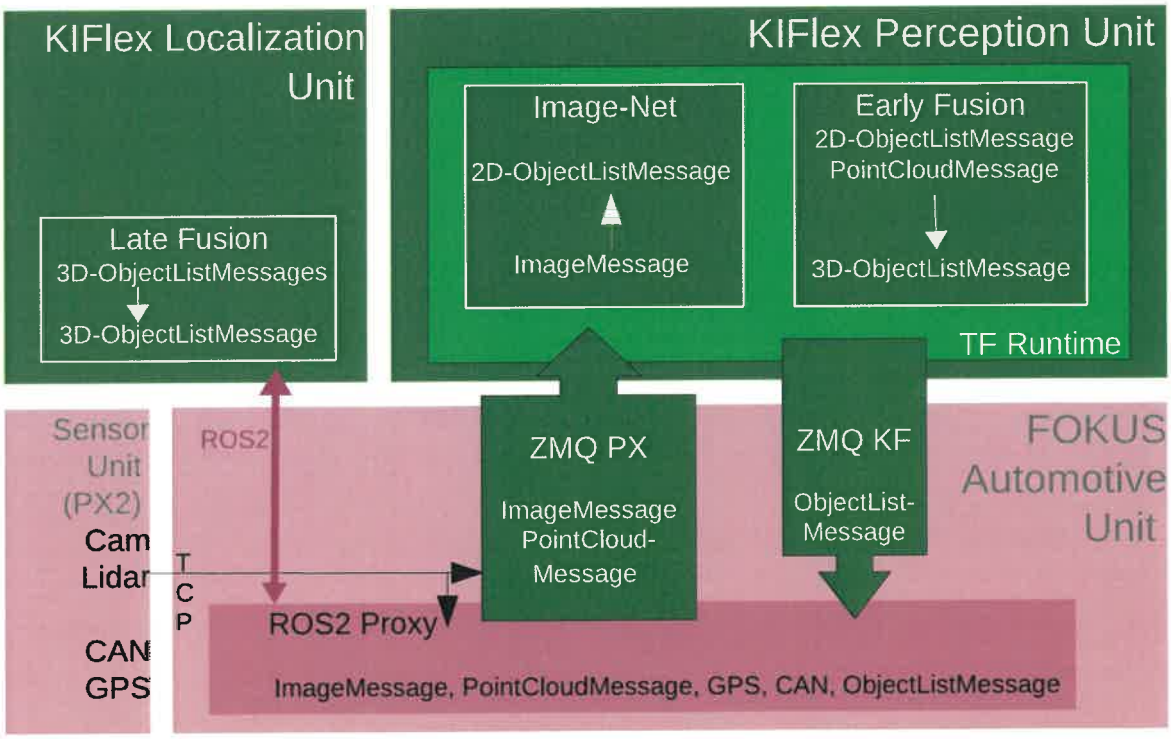
Der Entwurf des Sensor-Datenfusion-Softwareframeworks wurde durch FOKUS erfolgreich realisiert und umfasst die Komponenten zur Kommunikation der Sensordaten, der Detektion der 2D-Objekte und schließlich der Early Fusion von 2D-Detektionen und 3D-Lidar-Punkten (s. Abb. unten).

Die Sensordaten der Sensor Unit werden vom ROS2 Proxy der Automotive Unit empfangen und mittels einer effizienten Zero MQ-Verbindung durch den ZMQ PX-Sender an die Perception Unit des KIFLEX Systems weitergegeben. Die Sensordaten sind dabei in Messages der folgenden Typen kodiert:

- **ImageMessage**: komprimierte Bildinformation der Kameras (30 Hz)
- **PointCloudMessage**: 3D-Punktwolke der Lidardaten
- **ObjectListMessage**: Liste der detektierten und/oder (early) fusionierten Objekte, als Resultat der Detektion der Umfeldwahrnehmung und der Early Fusion

Die ImageMessage gelangt zuerst zum **Image-Net** der Detektion der Umfeldwahrnehmung. Hier erfolgt mittels Deep Learning Netze eine Detektion der 2D-Bounding Boxen der Objekte im Kamerabild.

Das Detektionsresultat (2D-ObjectListMessage) und die 3D-Punktwolke der Lidardaten (PointCloud) bilden den Input des Early Fusions Moduls. Das Fusionsergebnis ist die 3D-ObjectListMessage mit der der 3D-Position und Ausdehnung der detektierten Objekte.



Die 3D-ObjectListMessage des Early Fusion Moduls wird über den ZMQ KF Sender zurück an den ROS2 Proxy der Automotive Unit gegeben und hier in eine allgemein verfügbare ROS2-Message umgewandelt. Diese Resultat-Message der Early Fusion ist so auch in der KIFLEX Localization Unit verfügbar und kann vom Late Fusion Modul (IFAG) weiterverarbeitet werden.

Die ObjektListMessage (ROS2) als Resultat der Early Fusion ist dabei wie folgt spezifiziert:

ObjectListMessage

Object message containing the list of the detected 2D/3D Object.

ATTRIBUTES

std_msgs/Header header

Standard ROS header with timestamp

Object[] objects

List of objects

Object

Object message containing the detected 2D/3D Object.

Available object types:

uint32 PERSON = 24
uint32 CAR = 26
uint32 BICYCLE = 33

status
uint32 VALID = 0
uint32 INVALID = 1

ATTRIBUTES

std_msgs/Header header

Standard ROS header with timestamp

uint32 id

Id of the object or 0

uint32 type

object cass (label)

string subtype

label for subcategory, e.g. traffic sign id

float32 type_confidence

confidence of the object type (between [0, 1])

float32 subtype_confidence

confidence of the object subtype (between [0, 1])

Size size

length, width, height

Point relative_position

in [m], x, y, z

Orientation relative_orientation

in [rad], yaw

UTMPose absolute_pose

UTM position and heading

uint32 id Object2D object2d

image of the object, position and size in image coordinates

uint32 status

Status of object

Bool invalid

Object2D

Object message containing the detected 2D Object.

ATTRIBUTES

std_msgs/Header header

Standard ROS header with timestamp

uint32 x

position x in [px], center of the object in the original image

uint32 y

position y in [px], center of the object in the original image

uint32 width

width in [px]

uint32 height

height in [px]

sensor_msgs/Image image

AP 2.5 Algorithmen-Entwicklung

AP 2.5.1 Entwicklung der KI-basierten Umfeldwahrnehmung

In diesem AP entwickelte FOKUS das Softwareframework zur Umfeldwahrnehmung durch Detektion von Verkehrsobjekten mittels Deep Learning Netzwerken. Das AP wurde erfolgreich abgeschlossen. Die integrierenden Arbeiten erfolgten im AP der Gesamtintegration des Perzeptions-Softwareframeworks.

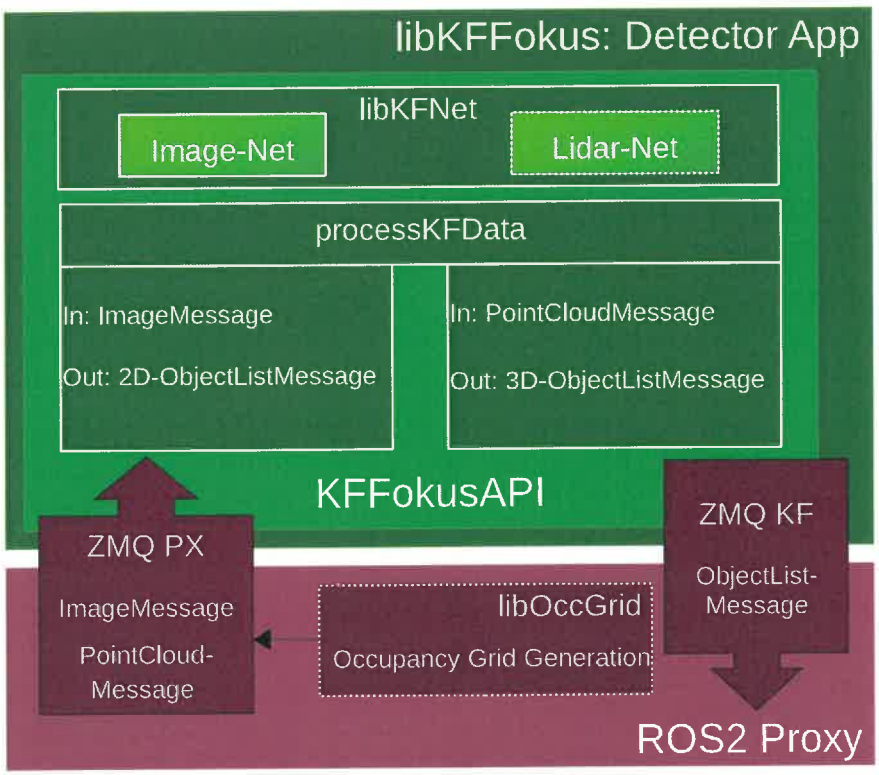
Die Umfeldwahrnehmung realisiert Detektion der Umgebungobjekte mittels der Deep Learning Netze aus AP 2.5.4 in den Kamerabildern, Image-Net, und den 3D-Punktwolken der Lidarsensoren, Lidar-Net.

Der Entwurf des Softwareframeworks der Umfeldwahrnehmung wurde durch FOKUS erfolgreich realisiert und umfasst die Komponenten zur Kommunikation der Sensordaten, der Detektion der 2D-Objekte und 3D-Objekte (s. Abb. unten).

Die Sensordaten der Sensor Unit werden vom ROS2 Proxy der Automotive Unit empfangen und mittels einer effizienten Zero MQ-Verbindung durch den ZMQ PX-Sender an die Perception Unit des KIFLEX Systems weitergegeben. Die Sensordaten sind dabei in Messages der folgenden Typen kodiert:

- **ImageMessage**: komprimierte Bildinformation der Kameras (30 Hz)
- **PointCloudMessage**: 3D-Punktwolke der Lidardaten
- **ObjectListMessage**: Liste der detektierten 2D-Objekte (Image-Net), 3D-Objekte (Lidar-Net)

Die ImageMessage gelangt zuerst zum **Image-Net** der Detektion der Umfeldwahrnehmung. Hier erfolgt mittels Deep Learning Netze eine Detektion der 2D-Bounding Boxen der Objekte im Kamerabild.
Die 3D-Punktwolke der Lidardaten (PointCloud) wird durch das Lidar-Net verarbeitet, das von FOKUS konzipiert, implementiert und in die Umfeldwahrnehmung integriert wurde. Sein Detektionsresultat ist die 3D-ObjectListMessage mit der der 3D-Position, Ausdehnung und Orientierung der detektierten Objekte.



Alle Hauptkomponenten der Umfeldwahrnehmung wurden in Form von C++ Libraries implementiert und den Partner zur Integration zur Verfügung gestellt. Die Integration des Systems in das Perceptions-Softwareframework des Projektes erfolgte im Integrations-AP.

Das System der Umfeldwahrnehmung umfasst die bereits erfolgreich realisierten Komponenten:

- **ROS2 Proxy:** Umwandlung der Sensordaten in ROS2 Messages, Distribution der Daten an alle Komponenten des gesamten KIFLEX Softwareframeworks, Empfang der Detektionsresultate der Deep Learning Netze des Umfeldsystems und Umwandlung in ROS2 Messages. Der ROS2 Proxy ist als ROS2-Node implementiert.

- **ZMQ PX:** ZMQ-Sender der Sensordaten der Kamera und des Lidars als ZMQ-Messages: ImageMessage, PointCloudMessage. C++ Modul.

- **KFFokusAPI:** Definition der Funktionalität des Softwareframeworks der Umfeldwahrnehmung zur Integration ins KIFLEX-Softwareframework. C++ API.

  processKFData: Haupverarbeitungroutine der Umfeldwahrnehmung zum Pre- und Postprocessing der Daten und zum Aufruf der Inference-Calls der DL-Netze.

- **libKFFokus:** Implementierung der Basisfunktionalitäten des Softwareframeworks der Umfeldwahrnehmung zur Kommunikation, Parallelverarbeitung und Prozeßsteuerung.

- **Detector App:** ausführbare Realisierung des Systems der Umfeldwahrnehmung in der Tensorflow Runtime der KIFLEX Perception Unit. C++ Executable.

- **libKFNet:** C++ Anbindung des Softwareframework der Umfeldwahrnehmung an die C++ API der Tensorflow Runtime. Realisiert die Inintialisierung und Inference-Call der Deep Learning Netze. C++ Library.

- **Image-Net:** von FOKUS entwickelte Variante eines Inception V2 SSD Deep learning-Netzes zur Dektektion von Verkehrsobjekten im Kamera-Bild zur Umfeldwahrnehmung. Tensorflow-Netzwerk.

- **libOccGrid:** C++ Implementierung der Erzeugung und Kompression von Occupancy-Images aus den 3D-Punktwolken der Lidar-Daten als Input des Lidar-Net der Umfeldwahrnehmung. C++ library.

- **Lidar-Net:** von FOKUS entwickeltes Deep Learning-Netz zur Dektektion von Verkehrsobjekten allein auf Basis von Lidar-Daten zur Umfeldwahrnehmung. Das Netzwerk detektiert dabei die Klasse, 3D-Position, Abmessung und Orientierung des Umfeldobjektes (bisherig Tests: nur die Klasse: Car). Tensorflow-Netzwerk.

Weitere Informationen zum Stand des Trainings und Tests der Netze ist in AP 2.5.13/14 angegeben.

AP 2.5.2/3 Entwicklung des Datenfusionsmoduls (Early Fusion)

Im AP wurde von FOKUS das Early Fusion Modul zur frühen Datenfusion, in dem die 2D-ObjectListMessages der Umfeldwahrnehmung mit den 3D-Punktwolken der Lidare zu 3D-ObjectListMessages kombiniert werden, entwickelt.

In AP 2.5.3 entwickelt IFAG das Late Fusion Modul, das die 3D-ObjectListMessages des Early Fusion Moduls mit weiteren Sensordaten auf Objektebene fusioniert, und wird hierbei durch FOKUS in der Integration der Resultate des Early Fusion Moduls unterstützt.

Der Entwurf des Early Fusion Moduls wurde durch FOKUS erfolgreich realisiert und umfasste die Komponenten zur Kommunikation der Sensordaten und der Fusion der detektierten 2D-Objekte der Umfeldwahrnehmung aus AP 2.5.1 und 3D-Punktwolken der Lidar-Sensoren.

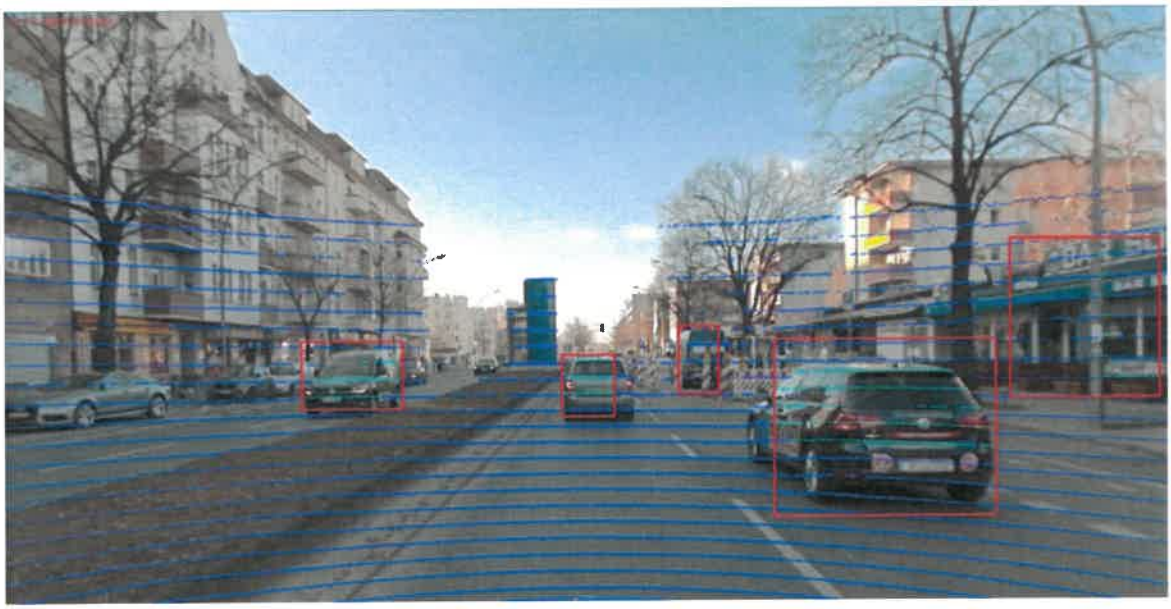
Die Sensordaten der Sensor Unit werden vom ROS2 Proxy der Automotive Unit empfangen und mittels einer effizienten Zero MQ-Verbindung durch den ZMQ PX-Sender an die Perception Unit des KIFLEX Systems weitergegeben. Die Sensordaten sind dabei in Messages der folgenden Typen kodiert:

- **ImageMessage**: Kameradaten – Input für die Umfeldwahrnehmung AP 2.5.1
- **2D-ObjectListMessage**: Liste der detektierten 2D-Objekte der Umfeldwahrnehmung
- **PointCloudMessage**: 3D-Punktwolke der Lidardaten
- **3D-ObjectListMessage**: Liste der fusionierten 3D-Objekte des Early Fusion Moduls

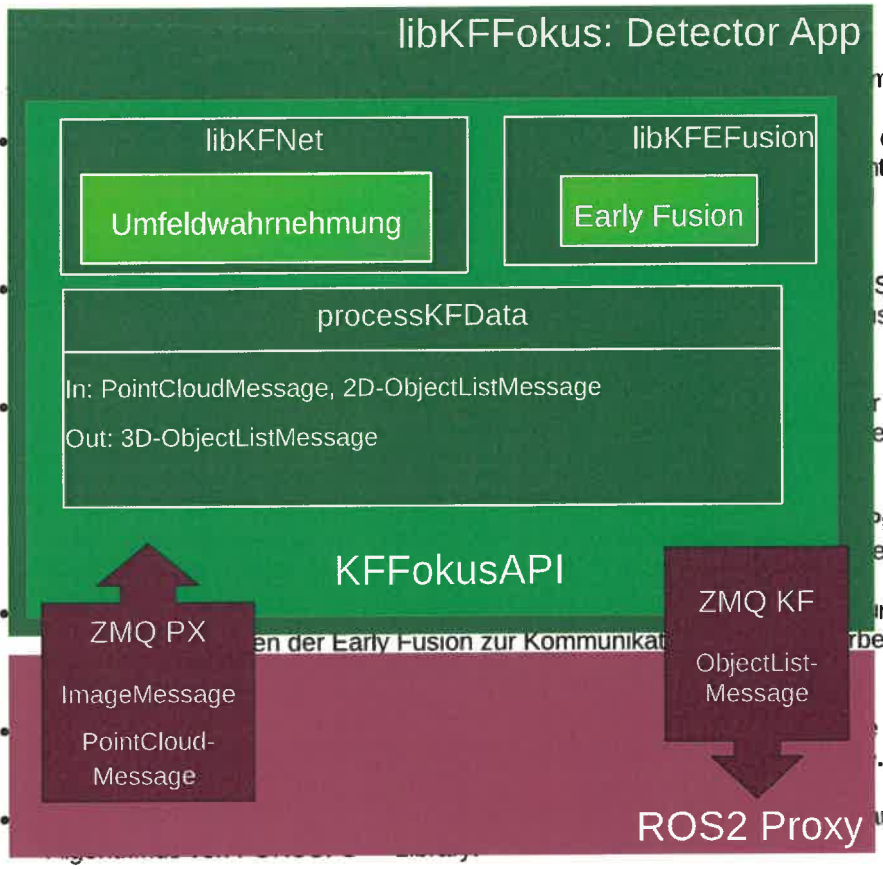
Die ImageMessage gelangt zuerst zur Umfeldwahrnehmung. Hier erfolgt mittels Deep Learning Netze eine Detektion der 2D-Bounding Boxen der Objekte im Kamerabild. Die 2D-ObjectListMessage und die PointCloudMessage der Lidar-Daten stellen den Input des Early Fusion Moduls dar. Sie werden durch den von FOKUS implementierten Early-Fusion-Algorithmus zu 3D-ObjectListMessages mit der 3D-Position und Abmessung der detektierten Umgebungsobjekte fusioniert.

Der von FOKUS im Projekt erfolgreich realisierte Early-Fusion-Algorithmus nutzt eine Projektionsmethode basierend auf der Anwendung der extrinsischen Transformation zwischen Kamera- und Lidar-Bezugssystem, sowie der projektiven Kamera-Abbildung in den u-v-Pixelspace des Image-Bezugssystems. Als Resultat erhält man die 2D-Positionen der 3D-Lidar-Punkte im Kamerabild (s. Abb. unten: Die blauen Punkte sind die Projektionen der 3D-Lidar-Punkte).

Nun können die 3D-Punktmengen (s. Abb.: hellblaue Punkte) innerhalb der 2D-Boxen der Objekt-Detektionen (s. Abb.: rote Boxen) bestimmt werden. Der Schwerpunkt und die Standardabweichung dieser Teilpunktmengen stellen eine Approximation für die 3D-Position und Abmessung der detektierten Umfeldobjekte dar.



Die Implementierung und der Test aller Hauptkomponenten (s. Abb.) der Early Fusion in Form von C++ Libraries erfolgreich bearbeitet und den Partner zur Integration zur Verfügung gestellt.



Das ...mponenten:

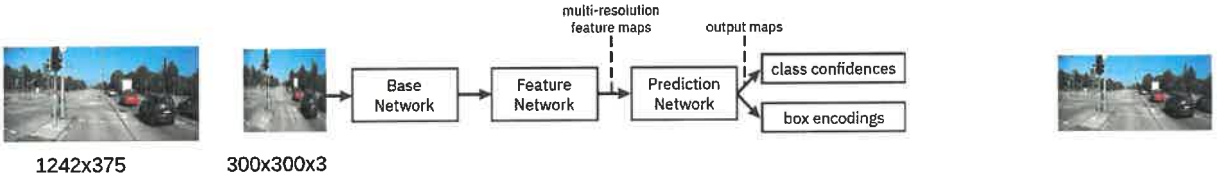
- ... der Sensordaten in ...ten KIFLEX ... Umwandlung in
- ... Sensordaten der ...sion als ZMQ-
- ... Funktionalität des ...eframework. C++
- ...Projektion und ...ekte.
- ...ng der ...beitung und
- ... Realisierung des ...
- ...rly-Fusion-

Die Implementierung des Early Fusion Softwareframeworks wurde abgeschlossen und erfolgreich getestet. Die Abb. oben zeigt eine Beispiel der Umfeldwahrnehmung und Early Fusion mit Realdaten aus dem Berliner Testgebiet des SAFARI-Projektes.

AP 2.5.4 Algorithmen-Entwicklung Detektion

In diesen AP hat FOKUS im erfolgreich Algorithmen zur Detektion von Objekten in der Umfeldwahrnehmung entwickelt, implementiert und getestet. Durch das Behave-Detektionsframework besitzt FOKUS Vorwissen, daß in die Anforderungsanalyse und den Entwurf der KIFLEX-Detektionsmodule eingebracht wurde. Das Behave-System ist einem Versuchsträger implementiert, der mittels Sensoren (Kameras und Lidar) Daten erfasst und Verkehrsobjekte dann mittels Deep Learning-Netze detektiert.

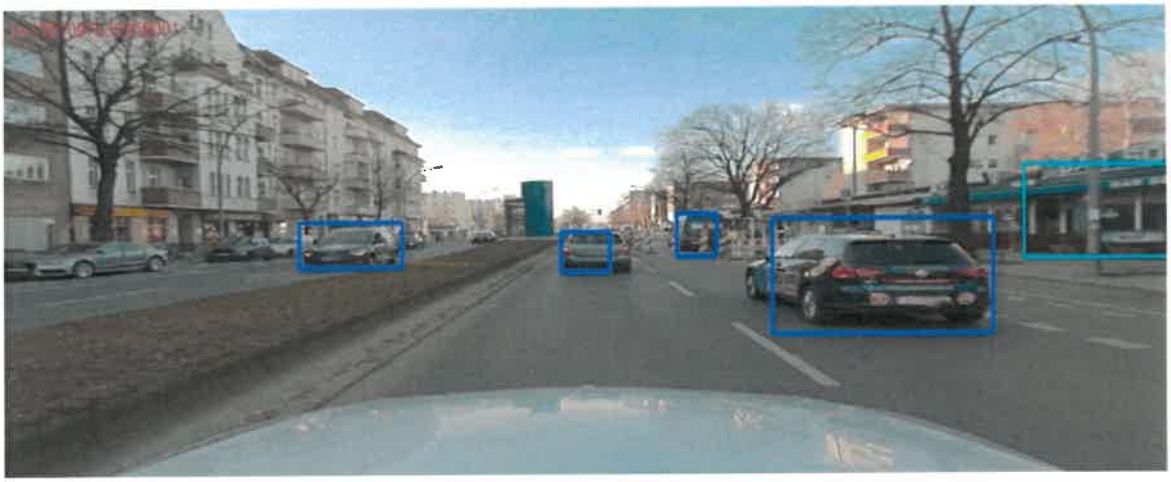
Die Netzwerkarchitektur eines Inception V2 SSD die sich hier bewährt hat, wurde als Entwurf in das Projekt eingebracht wurde. Eine Testversion dieses Netzes wurde den Partnern zur Analyse zur Verfügung gestellt.



Das DL-Netz kann zur Detektion in den Kamerabildern in Form der im Projekt entwickelten Tensorflow-Implementierung – Image-Net – und in zur Detektion in Occupancy-Images der Lidar-Daten – Lidar-Net – verwendet werden.

Die Implementierung, Integration und der Test des Image-Net wurde bereits im AP 2.5.2 bzw. 2.5.1 erfolgreich durch FOKUS realisiert. Das Lidar-Net ist in der derzeitigen Entwicklung in AP 2.5.1 und es wird Objekte direkt aus den Lidar-Daten detektieren. Erste Tests in AP 2.4.2 haben gezeigt, dass dazu ebenfalls ein Inception V2 SSD DL-Net jedoch mit einem größeren input layer (1000x1000) zu bevorzugen ist. Dieses Netzt wird direkt 3D-Posen (Position, Size, Orientierung) der Objekte erkennen können.

Die untere Abb. zeigt ein erstes Testergebnis des Image-Net mit Realdaten aus dem Berliner Testgebiet des SAFARI-Projektes.



Die blauen Boxen stellen Detektionen der Klasse „Car" dar. Es zeigen sich noch Fehldetektionen (hellblaue Box), die durch weiteres Training der Netze in AP 2.5.13 und 2.5.14 verbessert werden sollen.

AP 2.5.13/2.5.14 Training/Test der KI-Algorithmen

Die im AP erfolgreich realisierten Netze wurden im AP der Gesamtintegration in das Softwareframework des Projektes integriert. Im AP wurden von FOKUS die Deep Learning Netze aus AP 2.5.4 zur Detektion in der Umfeldwahrnehmung AP 2.5.1 trainiert und getestet.

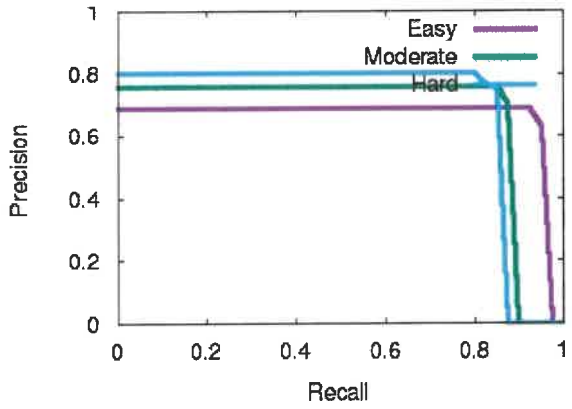
Als Trainings- und Referenz-Daten-Set fand der KITTI dataset Anwendung, der neben Image-Trainings-Daten (training 5236 images, Validation 2245 images) auch Lidar-daten (training 5236 lidar point clouds, Validation 2245 images) enthält.

Weiterhin kammen von FOKUS im SAFARI-Projekt aufgenommene Real-Daten aus dem Berliner SAFARI-Testgebiet zum Einsatz. Zu Bereitstellung eines Testsatz an die Partnern wurde eine DSGVO-Vereinbarung zu projektinternen Forschungszwecken initiiert.

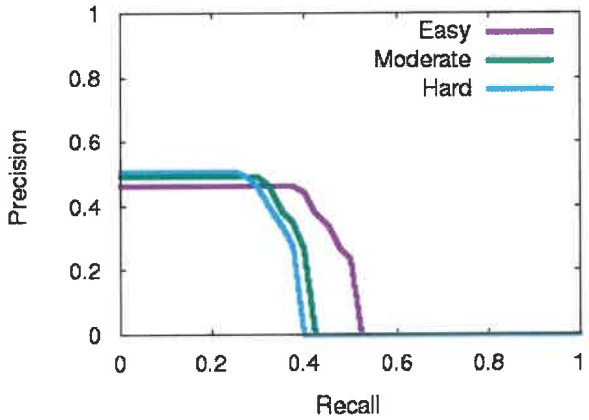
Resultate des Trainings und Tests im letzten Berichtzeitraum des Image-Net zeigten die gute Eignung dieser KI-Algorithmen.

mAP	Easy	Medium	Hard
Car	0.654	0.663	0.682
Pedestrian	0.221	0.192	0.184
Cyclist	0.381	0.361	0.348

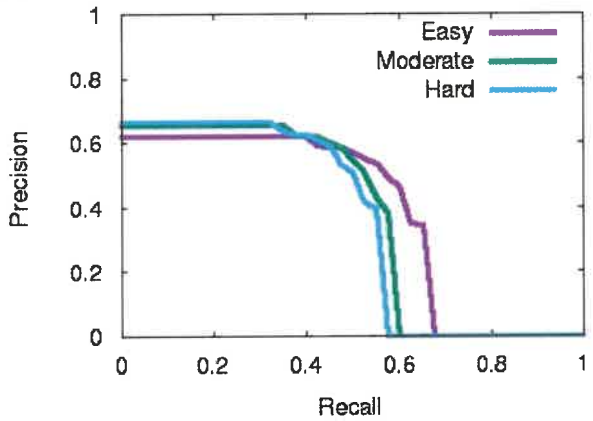
Car



Person



Cyclist



Es wurde daher angestrebt durch weiteres Training der Netze die Detektionsgüte (mean Average Precision) noch zu steigern. Dies wurde durch die Arbeiten von Herrn Sharma erbracht (s. Anhang A).

Zur Erstellung von geeigneten Datensätzen zum Trainieren der im Projekt entwickelten DNNs wurde das FOKUS-eigene Labeling-Tool **FLLT.AI** zur Verfügung gestellt:

FLLT.AI: KI-gestützte Annotation von 3D Punktwolkendaten

Die Verwendung von Künstlicher Intelligenz (KI) verbreitet sich aktuell immer schneller. Im Vergleich dazu sind die vorhandenen Datenquellen um solche Ansätze zu Trainieren sehr begrenzt. FLLT.AI ist ein KI-gestütztes Framework zur Annotation von 3-dimensionalen (3D) Punktwolkendaten, das vom Fraunhofer-Institut für Offene Kommunikationssysteme entwickelt wird. Zum Einen besteht es aus

einem Backend zum hochladen, speichern und (vor-)verarbeiten von 3D Punktwolkendaten und damit assoziierten Daten, wie Bildern, GPS Positionen und Sensor-Transformationsinformationen. Mit dem FLLT.AI Backend können hochgeladene Daten automatisch mit KI-gestützten Verfahren annotiert werden. Zum Anderen, stellt FLLT.AI ein Web-Annotationstool (FLLT.AI Labeling Tool) bereit, mit dem Daten inspiziert und editiert werden können. Dafür lädt das FLLT.AI Labeling Tool 3D Punktwolken- und Bilddaten aus dem Backend und visualisiert sie Seite-an-Seite und ermöglicht eine Korrelation der Daten durch Sensor-Transformationsinformationen. Das FLLT.AI Labeling Tool ermöglicht punktweise Klassen- und Instanzannotationen durch verschiedene Bearbeitungswerkzeuge und Ansichten.

FLLT.AI Backend

Das FLLT.AI Backend ist auf das Integrated Testing and Evaluation Framework (ITEF) aufgebaut, welches ebenfalls vom Fraunhofer-Institut für Offene Kommunikationssysteme entwickelt wird. Das Backend stellt Funktionen zur Datenverwaltung für 3D Punktwolkendaten zur Verfügung, die von Tiefensensoren zur Abstandsmessung (wie zum Beispiel einem light detection and ranging (LiDAR) Sensor) erzeugt wurden. Zusätzlich können assoziierte Daten ergänzend mit den 3D Punktwolkendaten hochgeladen und verwaltet werden. Diese assoziierten Daten können Fahrzeuggeschwindigkeiten und Positionen, aber insbesondere passende Kamerabilder des Umfelds, welches von den Tiefensensoren aufgezeichnet wurde, enthalten. Zusätzlich wird eine Spezifikation des Versuchsträgers verwendet um Kamera- und Tiefensensordaten zu korrelieren. Dies erlaubt nicht nur eine bessere Bearbeitung der Daten im FLLT.AI LabelingTool (siehe unten), sondern erweitert außerdem die Möglichkeiten zur automatisierten Datenverarbeitung. Denn das FLLT.AI Backend ist nicht nur eine passende Datenablage für Punktwolkendaten sondern ermöglicht auch die automatisierte Verarbeitung von Punktwolkendaten. Dies schließt eine Vor- und Nachverarbeitung ein um ein manuelles Annotieren mit dem FLLT.AI Labeling Tool effizienter und einfacher zu gestalten.

Im Backend werden Daten mit Hilfe von Szenarios organisiert. Jedes Szenario kann unterschiedlich konfiguriert werden, so dass verschiedene automatische Verarbeitungsschritte ausgeführt werden. Neu hochgeladene Daten werden automatisch in einen sogenannten Basket gruppiert. Dies wird im Backend registriert und kann je nach Konfiguration direkt erste Verarbeitungsschritte auslösen. Andere Ereignisse, wie das manuelle Annotieren von Daten, können weitere Verarbeitungsschritte hervorrufen. Dies ermöglicht individuelle Verarbeitungsstrategien umzusetzen. So können Daten zum Beispiel geschickt vorverarbeitet werden sobald sie hochgeladen wurden, dann können sie manuell annotiert werden mit dem FLLT.AI Labeling Tool und anschließend passend nachverarbeitet werden. Wenn eine manuelle Annotation nicht gewünscht ist, kann auch nur eine Abfolge von automatisierten Verarbeitungsschritten ausgeführt werden, nachdem die Daten hochgeladen wurden. Das FLLT.AI Labeling Tool kann dann immer noch zur Inspektion der Daten verwendet werden. Der aktuelle Verarbeitungsstatus kann in der Basketübersicht stets eingesehen werden.

Figure 1: FLLT.AI Backend-Datenorganisation: Szenarios und Baskets

FLLT.AI Labeling Tool

Das Labeling Tool des FLLT.AI Frameworks ist ein in den ITEF integriertes, browser-basiertes Tool. Wenn Sensordaten mit den zugehörigen Metainformationen wie Fahrzeuggeschwindigkeiten und -positionen hochgeladen werden, stellt FLLT.AI hervorragende Funktionen zur Datenexploration bereit. Mit Hilfe des ITEF können die Metainformationen visualisiert werden (zum Beispiel: Positionsverlauf des Versuchsträgers und seine Geschwindigkeiten auf einer Karte). Mit dem FLLT.AI Labeling Tool können die Sensordaten mit zugehörigen Bildern visualisiert werden. Der Hauptzweck des Labeling Tools ist jedoch die manuelle Annotation (Labeling) von 3D Punktwolkendaten. Von der Übersicht aller Daten aus kann ein Basket aus einem bestimmten Szenario gewählt und im Labeling Tool geöffnet werden. Dort kann dann jedes Sensordatum ausgewählt, visualisiert und editiert werden.

Im FLLT.AI Labeling Tool werden Daten nach sogenannten Frames organisiert. Für jedes Sensordatum existiert ein Frame, in dem zugehörige Daten wie Kamerabilder, Spezifikationen für Sensoren und Versuchsträger, sowie vorhandene Objekt-Detektionen korreliert werden. Um die Übersicht über Änderungen an einem Frame zu behalten, existiert eine Änderungshistorie für jeden Frame. Sie zeigt an wer die Daten hochgeladen hat, welche automatischen Verarbeitungsschritte an einem Frame vorgenommen wurden und wer manuelle Änderungen daran vorgenommen hat. Jeder Frame hat seine eigene Versionierung, es müssen also niemals Daten überschrieben werden, stattdessen wird eine neue Version eines Frames gespeichert. Die Änderungshistorie zeigt genau wer oder was eine bestimmte Version eines Frames erstellt hat.

Das primäre Konzept des FLLT.AI Labeling Tools basiert auf einer dualen Ansicht. Ein Frame wird dabei in zwei verschiedenen Ansichten nebeneinander dargestellt. Eine Ansicht basiert auf Kamerabildern, während die andere auf den 3D Punktwolkendaten eines Tiefensensors basiert. Beide Ansichten sind korreliert miteinander, indem die Punktwolkendaten in das Kamerabild projiziert werden. Diese duale Ansicht fördert das Szenenverständnis des Nutzers. Durch eine Hervorhebung der Punkte unter dem Cursor in der jeweils anderen Ansicht kann leicht festgestellt werden durch welche Oberfläche die Tiefeninformationen entstanden sind und somit das Labeling erheblich erleichtert werden.

Figure 2: FLLT.AI Labeling Tool: Duale Ansicht und Korrelation

Die Punktwolkenansicht stellt eine 2- und eine 3-dimensionale Ansicht bereit. In der 3D Ansicht ist die Kamera frei bewegbar innerhalb der Punktwolke. In der 2D Ansicht kann die Kamera entlang der vorgesehenen Ebene bewegt werden. Dabei kann aus vier verschiedenen Ebenen gewählt werden: in Richtung der Kamera, beide Seitenansichten 90 Grad versetzt zur Kamerasicht und eine Draufsicht. Die Annotationen der Punkte werden durch Farben kodiert. Weil gut unterscheidbare Farben begrenzt sind, kann zwischen verschiedenen Farbthemen gewählt werden. Das Bedienkonzept ist hauptsächlich auf die Tastatur ausgerichtet um effizient für professionelle Nutzer zu sein.

Das FLLT.AI Labeling Tool besitzt fünf Werkzeuge zur Bearbeitung der Punktwolkenannotationen. Zwei der Fünf Werkzeuge sind entworfen um Klassenannotationen pro Punkt zu bearbeiten. Sie funktionieren ähnlich wie die Mal-Werkzeuge einer Bildbearbungssoftware, mit dem Unterschied, dass nur Punkte der Wolke „angemalt" werden können. Diese beiden Werkzeuge besitzen unterschiedliche Formen und können in der Größe angepasst werden. Sie haben zwei Unterfunktionen integriert. Eine wählt die häufigste Klasse unter dem Werkzeug aus. Die Andere annotiert an allen Punkten unter dem Werkzeug die aktuell häufigste Klasse unter dem Werkzeug (Mehrheitsmodus).

Figure 3: FLLT.AI Labeling Tool: Ansichten und Farbthemen

Zwei weitere der fünf Werkzeuge sind zum Annotieren von Instanzen entworfen worden. Die Instanzen dienen der Unterscheidung verschiedener Objekte der gleichen Klasse. Diese Werkzeuge verwenden den gleichen konzeptionellen Ansatz wie die Werkzeuge zur Annotation der Klassen. Allerdings wird hier nicht eine Klasse zum Annotieren ausgewählt sondern entweder eine vorhandene Instanz von einem Objekt oder eine neue Instanz einer Klasse angelegt. Das Löschen von Instanzen ist eine Unterfunktion dieser Werkzeuge. Zusätzlich können alle Instanzen eines Frames angezeigt werden.

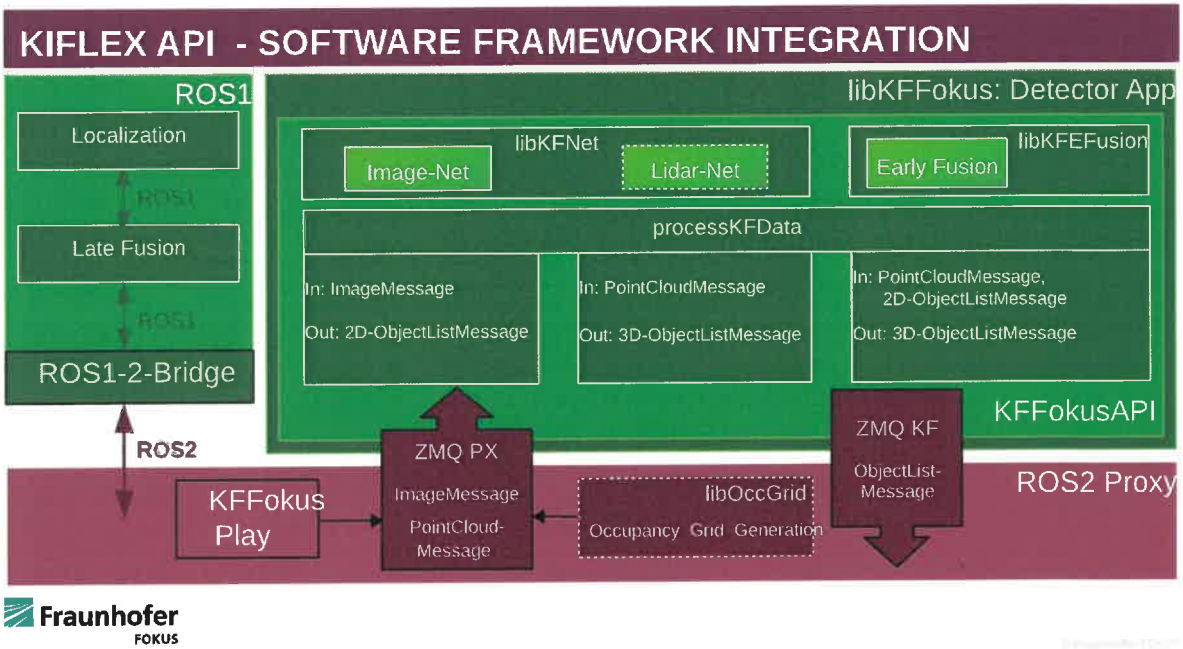
Wenn Instanzen annotiert werden, werden automatisch passende minimale Bounding Boxen generiert, beziehungsweise können Bounding Boxen durch Verarbeitungsschritte im FLLT.AI Backend hinzugefügt werden. Diese können mit Hilfe des fünften Werkzeugs inspiziert werden. Die Ausrichtung der automatisch erstellten minimalen Boxen kann zusätzlich mit dem Werkzeug editiert werden.

Neben diesen Hauptwerkzeugen existiert außerdem ein Werkzeug zum inspizieren der Reflektivitäten. Sie werden in der Regel von Laser-basierten Tiefensensoren erfasst und können Aufschluss über Materialbeschaffenheiten geben.

Um den Labeling Prozess noch effizienter zu machen existiert ein Filtersystem. Es ermöglicht momentan vier verschiedene Arten nach denen Punkte aus der Wolke temporär gefiltert werden können, um die entscheidenden Punkte einfacher Labeln zu können: Klasse / Kategorie, Radius, Höhe und Reflektivität. Zusätzlich unterstützt das Filtersystem spezielle Vorgehensweisen beim Labeln, wie zum Beispiel das Labeln von einer Klasse nach der anderen (fertig annotierte Klassen können dann einfach ausgefiltert werden).

AP 3.2 Konzeption des Sensor-Datenfusion-Frameworks

AP 3.2.1 SW-Framework-Integration



In diesen AP wurden die Umfeldwahrnehmung AP 2.5.1, die Early Fusion AP 2.5.2 und die Late Fusion AP 2.5.3, sowie die das ROS1 System der Localization Unit im KIFLEX Softwareframework planmäßig integriert. Die Einzelintegration der Hauptkomponenten: Detector App mit der KFFokusAPI und dem ROS2 Proxy der Automotive Unit konnte von FOKUS erfolgreich abgeschlossen und in das Gesamtsoftwareframework aller Hauptkomponenten: Detector App + ROS2 Automotive Unit + ROS1 Localization, integiert werden. Eine erste Validierung erfolgte im Rahmen der Demonstration des Projektes im Sommer 2022 auf dem Versuchsträger des DCAITI.

Hauptintegrationskomponente ist das von FOKUS implementierte ROS2 System der Automotive Unit. Hier werden alle Sensordaten und Detektions/Fusions-Resultate in ROS2-Messages transformiert und allen Systemkomponenten zur Verfügung gestellt. Diese Message-basierte Integration sorgt für eine sehr effiziente und einfache Integration aller Systemkomponenten bei hoher Verfügbarkeit und Flexibilität.

Dadurch konnte im AP die Integration folgender Hauptkomponenten im SW-Framework auf der Zielplatform ZCU102 erfolgreich realisiert:

KI-basierte Umfeldwahrnehmung

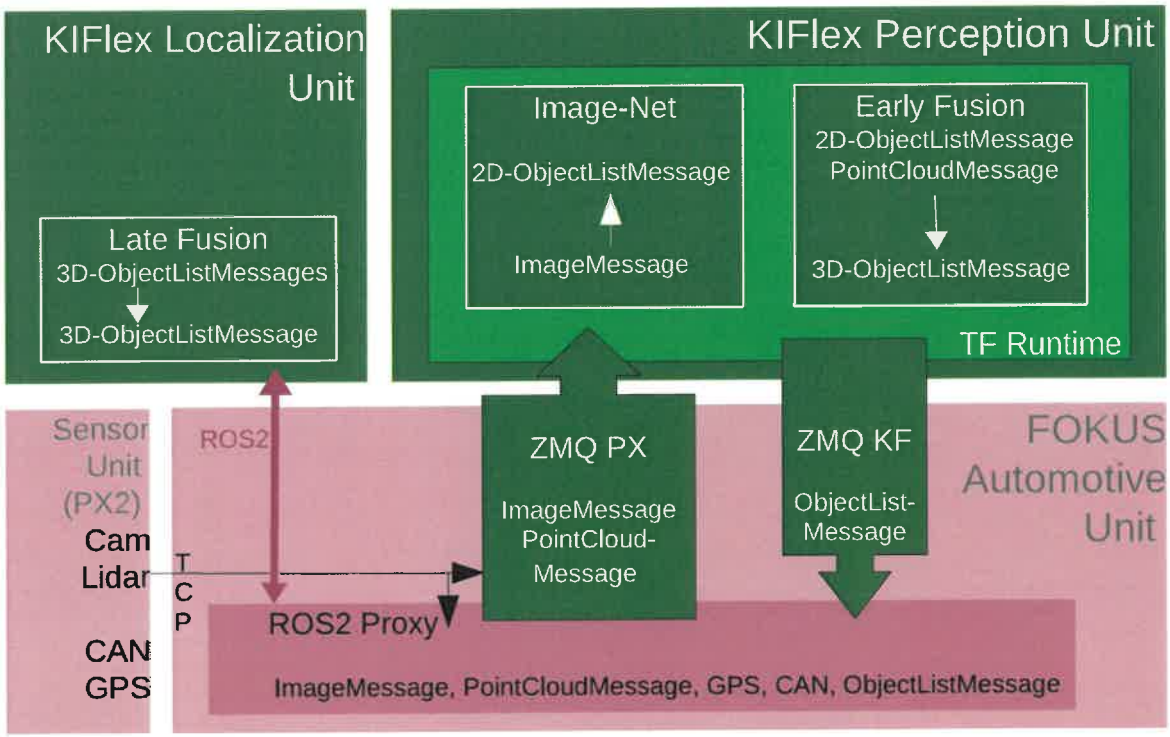
Die Komponente der Umfeldwahrnehmung (AP 2.4.2/2.5.1) detektiert Verkehrsobjekte, die in der Komponente des Fusions-Softwareframework zu 3D-Objekten fusioniert werden. Dabei unterscheidet man zwei Phasen der Fusion:

- Early Fusion Modul, Entwicklung FOKUS AP 2.5.2
- Late Fusion Modul, Entwicklung IFAG AP 2.5.3

Das Early Fusion Modul realisiert die frühe Phase der Fusion, in der die 2D-Objektdetektionen der Deep Learning Netze aus AP 2.5.4 in den Kamerabildern mit den 3D-Punktwolken der Lidarsensoren auf Low-Level-Ebene (Rohdaten) fusioniert werden. In Erweiterung dazu, entwickelte IFAG in AP 2.5.3 das Late Fusion Modul, das auf der High-Level-Ebene (Objekte) die verschieden Objektdetektionen fusioniert.

Das die von FOKUS entwickelte Umfeldwahrnehmung und Early Fusion nutzt die Komponenten der Kommunikation der Sensordaten, der Detektion der 2D-Objekte und schließlich der Early Fusion von 2D-Detektionen und 3D-Lidar-Punkten (s. Abb. unten).

Die Sensordaten werden über Zero MQ-Verbindungen an das **Image-Net** zur Detektion der 2D-Bounding Boxen der Objekte im Kamerabild weitergegeben. Die 2D-ObjectListMessage und die 3D-PointCloud werden im Early Fusions Modul zu 3D-ObjectListMessages mit der der 3D-Position und Ausdehnung der detektierten Objekte fusioniert. Die 3D-ObjectListMessage der Early Fusion ist als ROS2-Message für die KIFLEX Localization Unit und dem Late Fusion Modul (IFAG) verfügbar.



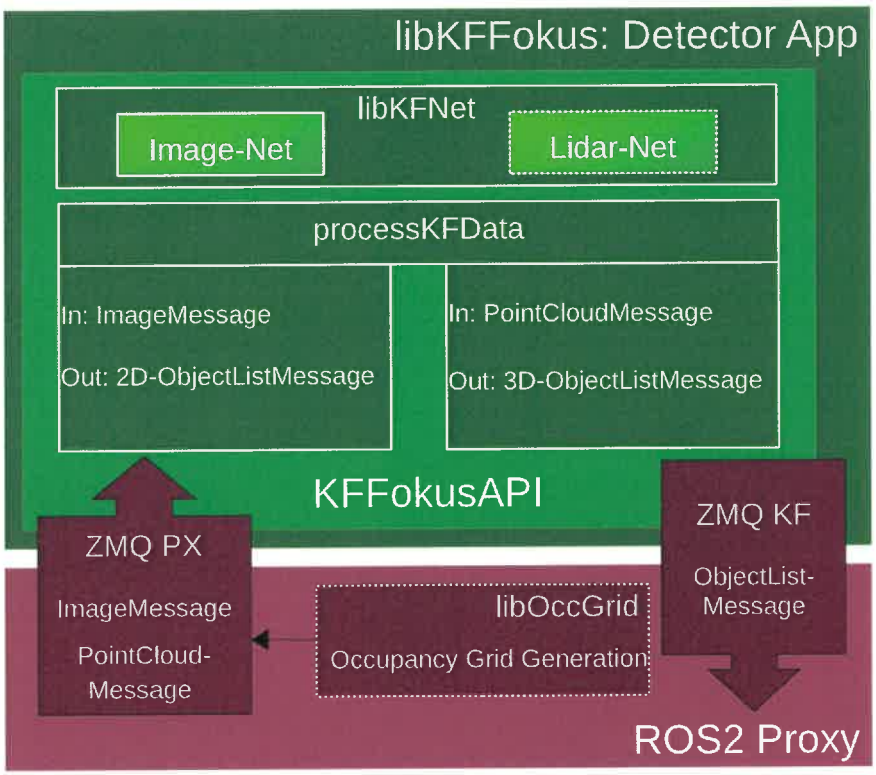
Die KI-basierte Umfeldwahrnehmung selbst realisiert dabei die Detektion der Umgebungobjekte mittels der Deep Learning Netze aus AP 2.5.4 in den Kamerabildern, Image-Net, und den 3D-Punktwolken der Lidarsensoren, Lidar-Net.

Die Integration des Softwareframeworks der Umfeldwahrnehmung umfasst die Komponenten zur Kommunikation der Sensordaten, der Detektion der 2D-Objekte und 3D-Objekte (s. Abb. unten).

Die Sensordaten der Sensor Unit werden vom ROS2 Proxy der Automotive Unit empfangen und mittels einer effizienten Zero MQ-Verbindung durch den ZMQ PX-Sender an die Perception Unit des KIFLEX Systems weitergegeben. Die Sensordaten sind dabei in Messages der folgenden Typen kodiert:

- **ImageMessage**: komprimierte Bildinformation der Kameras (30 Hz)
- **PointCloudMessage**: 3D-Punktwolke der Lidardaten
- **ObjectListMessage**: Liste der detektierten 2D-Objekte (Image-Net), 3D-Objekte (Lidar-Net)

Die ImageMessage gelangt zuerst zum **Image-Net** der Detektion der Umfeldwahrnehmung. Hier erfolgt mittels Deep Learning Netze eine Detektion der 2D-Bounding Boxen der Objekte im Kamerabild.
Die 3D-Punktwolke der Lidardaten (PointCloud) wird durch das Lidar-Net verarbeitet, das von FOKUS konzipiert, implementiert und in die Umfeldwahrnehmung integriert wurde. Sein Detektionsresultat ist die 3D-ObjectListMessage mit der der 3D-Position, Ausdehnung und Orientierung der detektierten Objekte.



Im Projekt wurden von FOKUS alle Hauptkomponenten der Umfeldwahrnehmung in Form von C++ Libraries in das Gesamt-SW-Framework integriert:

- **ROS2 Proxy:** Umwandlung der Sensordaten in ROS2 Messages, Distribution der Daten an alle Komponenten des gesamten KIFLEX Softwareframeworks, Empfang der Detektions/Fusionsresultate der Umfeldwahrnehmung/Early Fusion und Umwandlung in ROS2 Messages zur Weiterleitung an das ROS1 System der Localization Unit. Der ROS2 Proxy ist als ROS2-Node integriert.

- **ZMQ PX:** ZMQ-Sender der Sensordaten der Kamera und des Lidars als ZMQ-Messages: ImageMessage, PointCloudMessage. Als C++ Modul integriert.

- **KFFokusAPI:** Definition der Integration der Komponenten des KIFLEX-Softwareframework. C++ API.

- **libKFFokus:** Implementierung der Basisfunktionalitäten des Softwareframeworks der Umfeldwahrnehmung/Early Fusion zur Kommunikation, Parallelverarbeitung und Prozeßsteuerung. Als C++ Library integriert.

- **Detector App:** ausführbare Realisierung des Systems der Umfeldwahrnehmung in der Tensorflow Runtime der KIFLEX Perception Unit. C++ Executable.

- **libKFNet:** C++ Anbindung des Softwareframework der Umfeldwahrnehmung an die C++ API der Tensorflow Runtime. Realisiert die Inintialisierung und Inference-Call der Deep Learning Netze. Als C++ Library integriert.

- **Image-Net:** von FOKUS entwickeltes Deep Learning-Netz zur Dektektion von Verkehrsobjekten allein auf Basis von Kamerabildern zur Umfeldwahrnehmung. Das Netzwerk detektiert dabei die Klasse, 2D-Position und Abmessung des Umfeldobjektes. Tensorflow-Netzwerk.

- **libOccGrid:** C++ Implementierung der Erzeugung und Kompression von Occupancy-Images aus den 3D-Punktwolken der Lidar-Daten als Input des Lidar-Net der Umfeldwahrnehmung. C++ library.

- **Lidar-Net:** von FOKUS entwickeltes Deep Learning-Netz zur Dektektion von Verkehrsobjekten allein auf Basis von Lidar-Daten zur Umfeldwahrnehmung. Das Netzwerk detektiert dabei die Klasse, 3D-Position, Abmessung und Orientierung des Umfeldobjektes (bisherig Tests: nur die Klasse: Car). Tensorflow-Netzwerk.

Early Fusion Modul

Im AP wurde von FOKUS das Early Fusion Modul zur frühen Datenfusion, in dem die 2D-ObjectListMessages der Umfeldwahrnehmung mit den 3D-Punktwolken der Lidare zu 3D-ObjectListMessages kombiniert werden, integriert.

In AP 2.5.3 entwickelte IFAG das Late Fusion Modul, das die 3D-ObjectListMessages des Early Fusion Moduls mit weiteren Sensordaten auf Objektebene fusioniert, und wurde hierbei durch FOKUS in der Integration der Resultate des Early Fusion Moduls unterstützt.

Die Implementation des Early Fusion Moduls wurde durch FOKUS erfolgreich realisiert und umfasst die Komponenten zur Kommunikation der Sensordaten und der Fusion der detektierten 2D-Objekte der Umfeldwahrnehmung aus AP 2.5.1 und 3D-Punktwolken der Lidar-Sensoren.

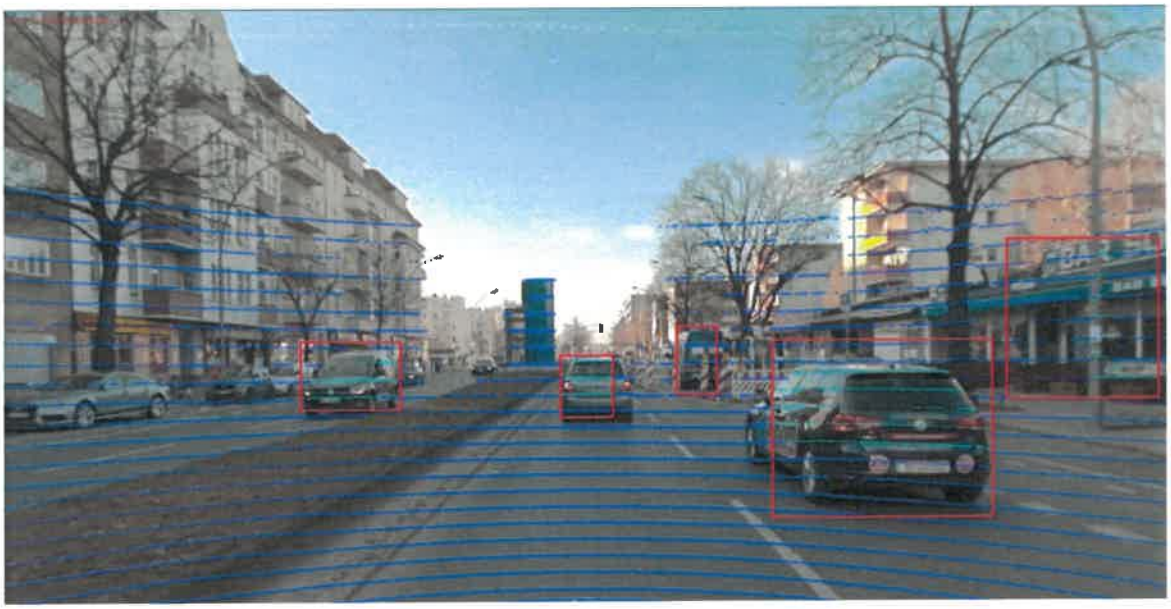
Die Sensordaten der Sensor Unit werden vom ROS2 Proxy der Automotive Unit empfangen und mittels einer effizienten Zero MQ-Verbindung durch den ZMQ PX-Sender an die Perception Unit des KIFLEX Systems weitergegeben. Die Sensordaten sind dabei in Messages der folgenden Typen kodiert:

- **ImageMessage**: Kameradaten – Input für die Umfeldwahrnehmung AP 2.5.1
- **2D-ObjectListMessage**: Liste der detektierten 2D-Objekte der Umfeldwahrnehmung
- **PointCloudMessage**: 3D-Punktwolke der Lidardaten
- **3D-ObjectListMessage**: Liste der fusionierten 3D-Objekte des Early Fusion Moduls

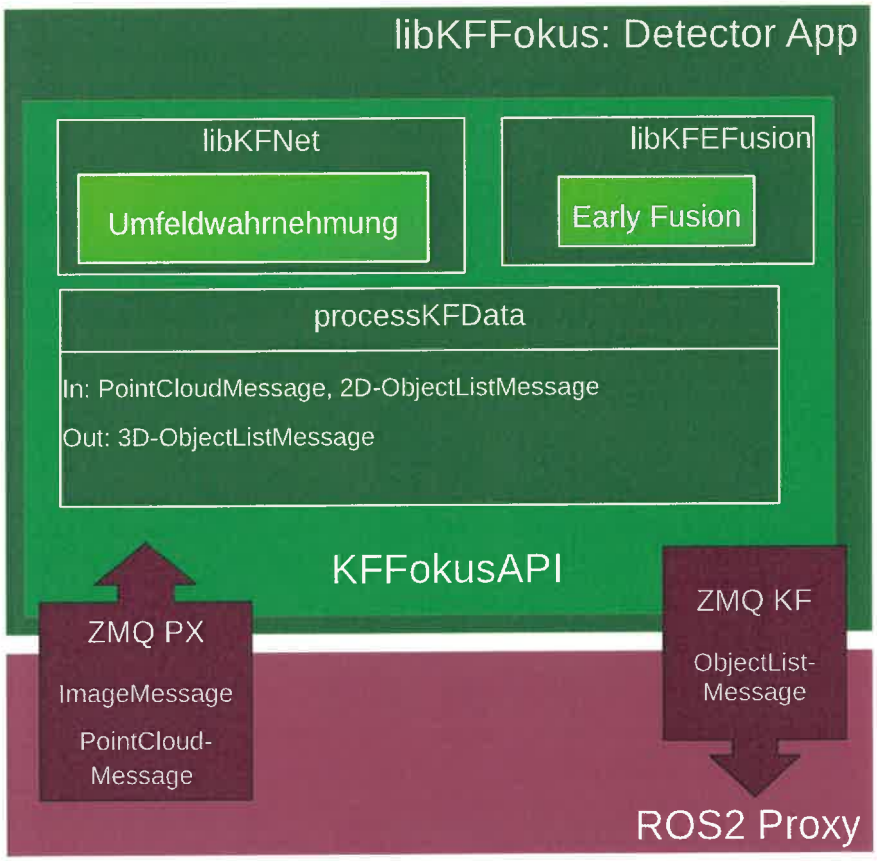
Die 2D-ObjectListMessage der Umfeldwahrnehmung und die PointCloudMessage der Lidar-Daten stellen den Input des Early Fusion Moduls dar. Sie werden durch den von FOKUS implementierten Early-Fusion-Algorithmus zu 3D-ObjectListMessages mit der 3D-Position und Abmessung der detektierten Umgebungsobjekte fusioniert.

Der von FOKUS erfolgreich integrierte Early-Fusion-Algorithmus nutzt eine Projektionsmethode basierend auf der Anwendung der extrinsischen Transformation zwischen Kamera- und Lidar-Bezugssystem, sowie der projektiven Kamera-Abbildung in den u-v-Pixelspace des Image-Bezugssystems. Als Resultat erhält man die 2D-Positionen der 3D-Lidar-Punkte im Kamerabild (s. Abb. unten: Die blauen Punkte sind die Projektionen der 3D-Lidar-Punkte).

Nun können die 3D-Punktmengen (s. Abb.: hellblaue Punkte) innerhalb der 2D-Boxen der Objekt-Detektionen (s. Abb.: rote Boxen) bestimmt werden. Der Schwerpunkt und die Standardabweichung dieser Teilpunktmengen stellen eine Approximation für die 3D-Position und Abmessung der detektierten Umfeldobjekte dar.



Die Integration aller Hauptkomponenten (s. Abb.) der Early Fusion wurde in Form von C++ Libraries erfolgreich realisiert.



Das Softwareframework der Early Fusion umfasst die von FOKUS realisierten Komponenten:

- **ROS2 Proxy:** auch Teil der Umfeldwahrnehmung AP 2.5.1. Umwandlung der Sensordaten in ROS2 Messages, Distribution der Daten an alle Komponenten des gesamten KIFLEX Softwareframeworks, Empfang der Fusionsresultate der Early Fusion und Umwandlung in ROS2 Messages. Der ROS2 Proxy ist als ROS2-Node integriert.

- **ZMQ PX:** auch Teil der Umfeldwahrnehmung AP 2.5.1. ZMQ-Sender der Sensordaten der Kamera für die Umfeldwahrnehmung und der Lidar-Daten für die Early Fusion als ZMQ-Messages: ImageMessage, PointCloudMessage. Als C++ Modul integriert.

- **KFFokusAPI:** auch Teil der Umfeldwahrnehmung AP 2.5.1. Definition der Funktionalität des Softwareframeworks der Early Fusion zur Integration ins KIFLEX-Softwareframework. C++ API.

  processKFData: Haupverarbeitungroutine des Early Fusion Moduls zur Projektion und Clusterbestimmung der 3D-Position und Abmessung der detektierten Objekte.

- **libKFFokus:** auch Teil der Umfeldwahrnehmung AP 2.5.1. Integration der Basisfunktionalitäten der Early Fusion zur Kommunikation, Parallelverarbeitung und Prozeßsteuerung.

- **Detector App:** auch Teil der Umfeldwahrnehmung AP 2.5.1. Ausführbare Realisierung des Systems der Umfeldwahrnehmung und der Early Fusion. Als C++ Executable integriert.

- **libKFEFusion:** C++ Implementation des Early Fusion Moduls mit dem Early-Fusion-Algorithmus von FOKUS. Als C++ Library integriert.

Die Integration des Early Fusion Softwareframeworks wurde abgeschlossen und erfolgreich getestet. Die Abb. oben zeigt eine Beispiel der Umfeldwahrnehmung und Early Fusion mit Realdaten aus dem Berliner Testgebiet des SAFARI-Projektes.

AP 3.3 Gesamt-System-Integration

AP 3.3.10 Gesamt-System-Validierung

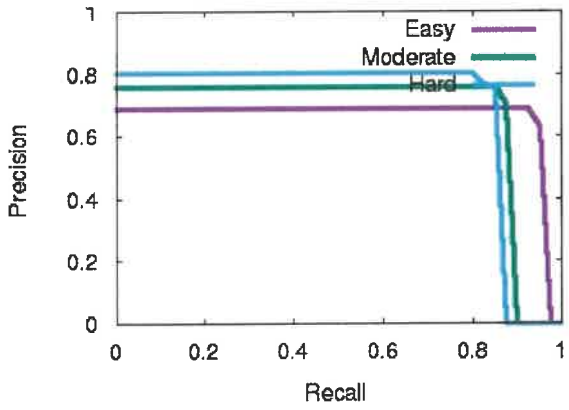
Im AP wurden von FOKUS die Deep Learning Netze aus AP 2.5.4 zur Detektion in der Umfeldwahrnehmung AP 2.5.1 in ihrer Integration im Gesamtsystem validiert.

Als Validierungs-Daten-Set fand der KITTI dataset Anwendung (Validation 2245 images, 5236 lidar point clouds). Weiterhin kammen von FOKUS im SAFARI-Projekt aufgenommene Real-Daten aus dem Berliner SAFARI-Testgebiet zum Einsatz. Zu Bereitstellung eines Testsatz an die Partnern wurde eine DSGVO-Vereinbarung zu projektinternen Forschungszwecken initiiert.

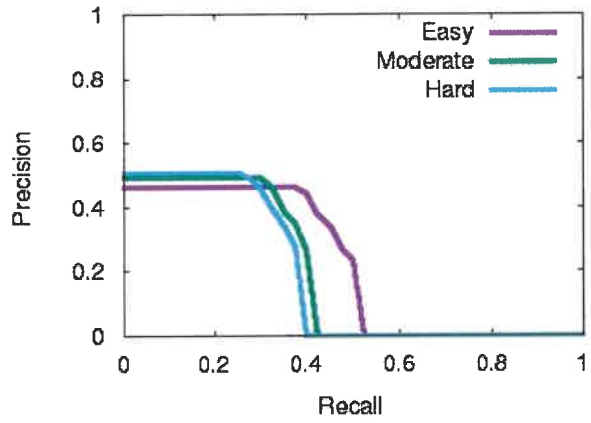
Die Resultate der Validierung des Image-Net zeigten die gute Eignung dieser KI-Algorithmen:

mAP	Easy	Medium	Hard
Car	0.654	0.663	0.682
Pedestrian	0.221	0.192	0.184
Cyclist	0.381	0.361	0.348

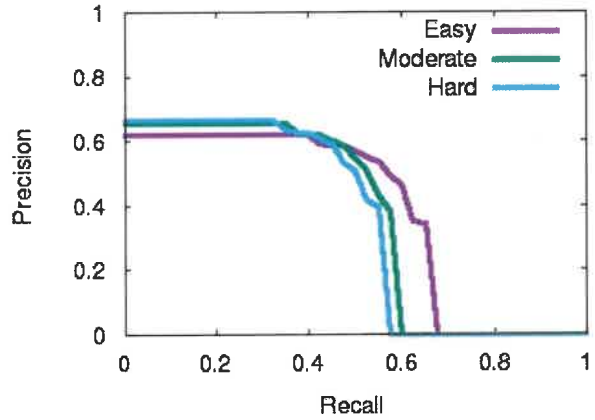
Car



Person

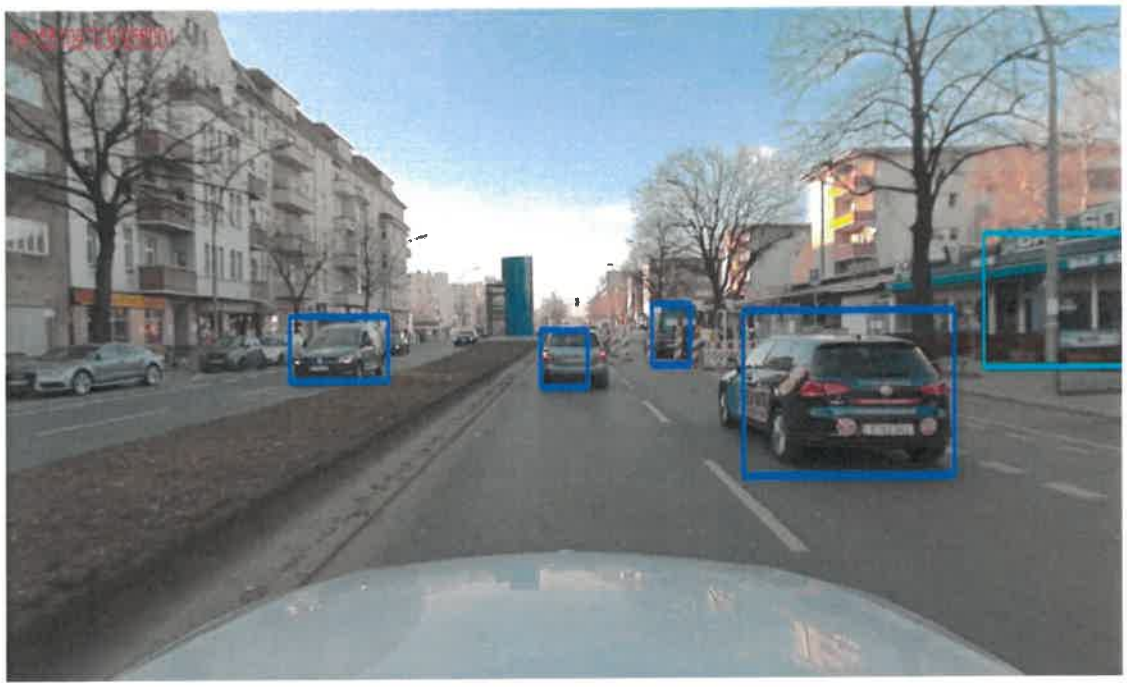


Cyclist



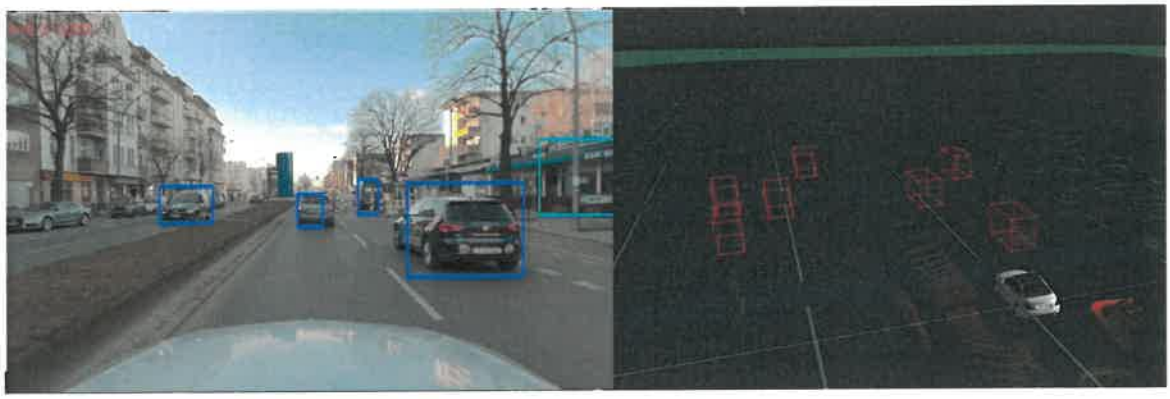
Einen genauen Überblick über die Validierung (wie auch der Konzeption und des Trainings) der KI-Netze gibt die Arbeit von Herrn Sharma im Anhang.

Die Validierung des Imagenet im Gesamtsystem des KIFLEX Softwareframework wurde anhand der eingefahrenen Daten des Safai-Projektes durch Vergleich der Kamerabilder und der Detektionen des Imagenet durchgeführt (s. Abb.).

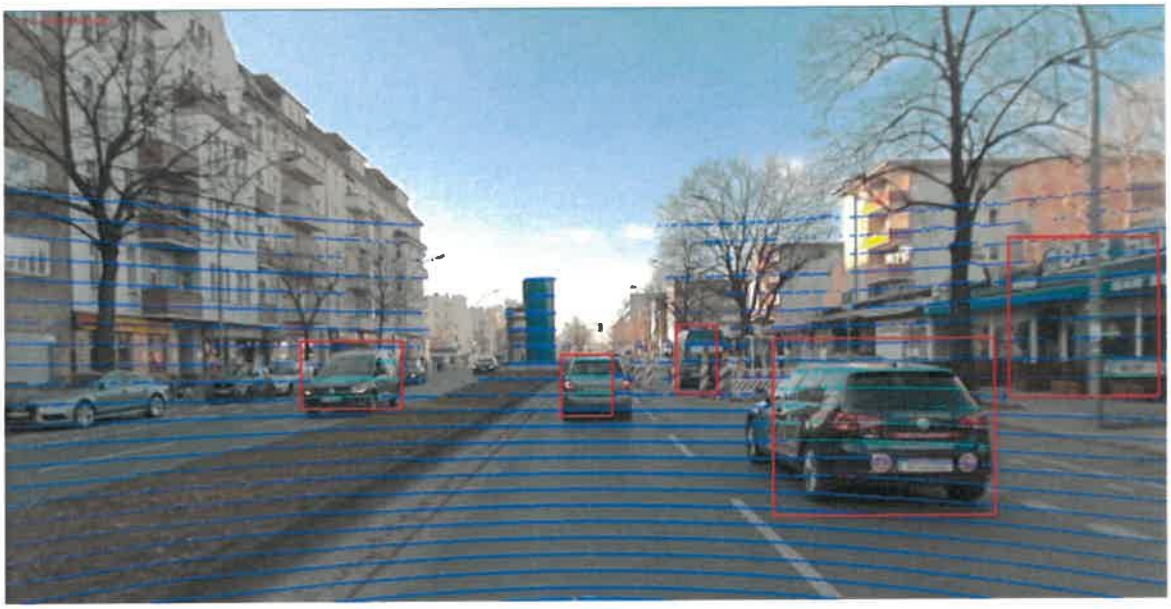


Das Lidarnet basisiert auf einer Detektion im Occupancy Grid der Punktwolke des Lidars. Diese wurde durch Vergleich mit den Kamerabildern und den Detektionen des Imagenet cross-validiert (s. Abb.).

Ebenso wurde die Validierung des Lidarnet im Gesamtsystem des KIFLEX Softwareframework anhand der eingefahrenen Daten des Safai-Projektes durch Vergleich der entsprechenden Kamerabilder und 2D-Detektionen des Imagenet mit den 3D-Detektionen des Lidarnet durchgeführt (s. Abb.).

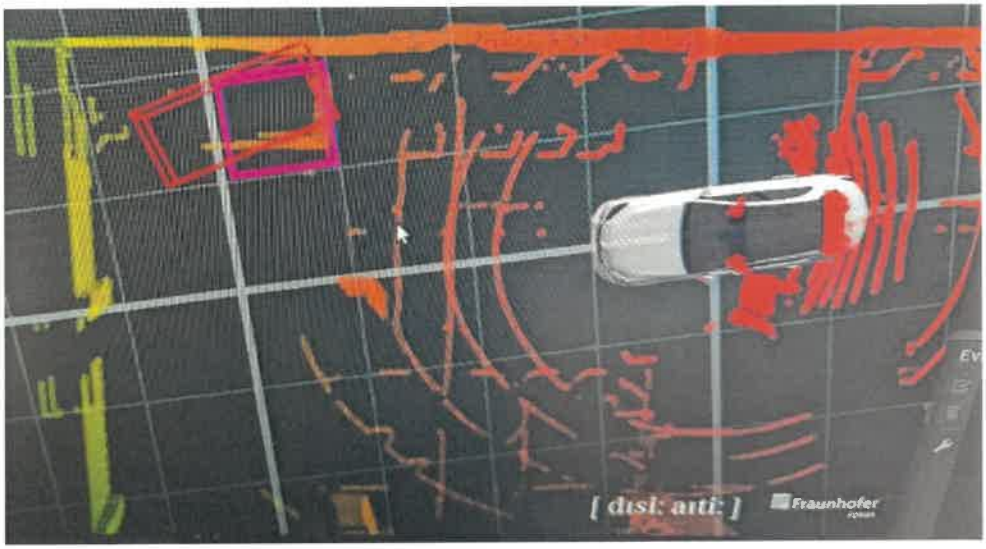


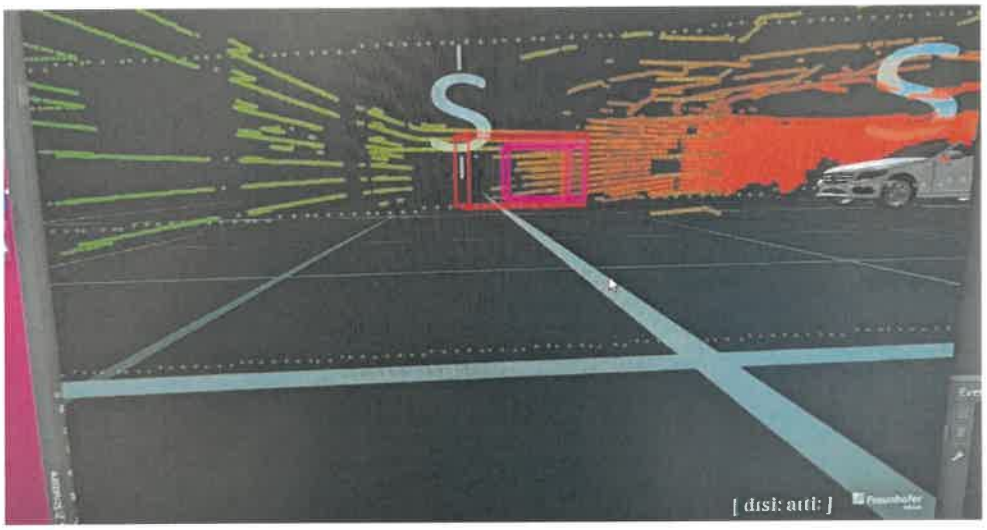
Das Early Fusion-Modul kombiniert die 2D-Detektionen und die Lidardaten durch einen Projektionsalgorithmus zu 3D-Detektionen. Die Cross-Validierung wurde dabei durch den Vergleich der Kamerabilder (s. Abb.), den 2D-Detektionen (rote Boxen) des Imagenet und den Lidar-3D-Daten (blaue Punkte) mit den projizierten Lidar-Punkten der Detektionen (hellblaue Punkte) realisiert.



Die Validierung des Gesamtsystem des KIFLEX Softwareframework wurde sowohl anhand der eingefahrenen Daten des Safai-Projektes durchgeführt (s. nächste Abb.), als auch auf der KIFLEX-Sommer-Demo live in Berlin gezeigt. Dabei wurden Real-Daten sowohl im Stand als auch in der Fahrt in Echtzeit auf dem Versuchsträger verarbeitet und die 3D-Detektionen visualisiert (s. folgende Abb.).


Media Playback Audio Video Subtitle Tools View Help
Fraunhofer FOKUS - DCAITI
Simulator View


[dɪsɪ: atɪ:]
Fraunhofer


[dɪsɪ: atɪ:]

3. Vergleich Planung und Umsetzung des Vorhabens

Arbeitsplanung

In 2020-22 kam es durch die COVID-19 Pandemie zu maßgeblichen Einflüssen auf die Arbeitswelt insbesondere auch für das Projekt. Zu Verzögerungen in der Bearbeitung und Fertigstellung von Arbeitspaketen im Gesamtprojekt sind ursächlich betragend für die Konzeptänderung des FlexAICore aufgrund der Nicht-Verfügbarkeit eFPGA-IP.

Zeitplanung

Zu Verzögerungen in der Bearbeitung und Fertigstellung von Arbeitspaketen im Teilprojekt FOKUS sind ursächlich betragend: Verzögerungen bei der Beschaffung von Hardware-Komponenten aufgrund von Störungen in den Lieferketten (z.B. bei Distributoren), Möglichkeiten zur Durchführung von Testfahrten mit Versuchsfahrzeugen waren eingeschränkt, sowie verzögerte Mitarbeiteranstellungen.

Daher wurde vom Gesamtprojekt eine Verlängerung beantragt.

Kostenplanung

Nicht-Verfügbarkeit bzw. Kostensteigerung von Bauteilen für ASIC-PCB erhöhten den Aufwand für das PCB-Design aufgrund dadurch notwendig werdender Änderungen und Suche nach Alternativbauteilen.

4. Ziele des Vorhabens

Zielerreichung

Die beschriebenen Verzögerungen und Lieferengpässe führten zu einer Konzeptänderung des FlexAICore und einer Projektverlängerung. Die vom FlexAICore unabhängige Arbeitspakete und insbesondere unabhängige Integrationsarbeiten zur Implementierung, Integration, Test und Demonstration des Softwareframeworks mit seinen KI-Netzen konnen zielgerecht realisiert werden.

Relevante F&E-Ergebnisse von dritter Seite mit Relevanz für die Durchführung des Vorhabens

Aktuell sind keine Ergebnisse von dritter Seite bekannt.

Änderung der Zielsetzung

Die beschriebenen Verzögerungen konnten im Rahmen der ursprünglich vorgesehenen Projektlaufzeit nicht ausgeglichen werden. Im Rahmen des Konsortiums wurde daher ein Antrag auf Projektverlängerung gestellt und eine Konzeptänderung des FlexAICore vollzogen.

Die genannten Einschränkungen führten im Rahmen des Gesamtprojektes zu folgenden Auswirkungen, die die Notwendigkeit einer Projektverlängerung begründeten:

- eingeschränkte Sammlung von Daten

- reduzierte Mitarbeiterkapazitäten
- Mitarbeiterverfügbarkeit zeitlich verschoben
- teilweise nur serielle Bearbeitung von Teilaufgaben möglich
- keine Erhöhung der Kapazitäten zum Ausgleich von Verzögerungen oder zur Parallelisierung von Arbeiten möglich
- Verschiebung der Bearbeitung und Fertigstellung von Teilaufgaben notwendig

Die initialen Einschränkungen und Auswirkungen der Corona-Pandemie führten zunächst zu einer Verschiebung des ASIC-Tapeout um 4 Monate von 08/2021 nach 12/2021. Die fortgesetzten Corona-Einschränkungen führten zur weiteren Verzögerung des RTL Code Freeze, wodurch ein zeitgrechter Abschluss des ASIC Backends auch bei 100% verfügbaren Ressourcen nicht mehr möglich war. Der Meilenstein "E7 - ASIC Tapeout" wurde damit um insgesamt 6 Monate verzögert. Ab 01/2022 war keine 100-prozentige Verfügbarkeit der benötigten Backend-Ressourcen gegeben.

5. Öffentlichkeitswirksame Maßnahmen

Es wurden im Projekt zwei Demonstartionen des im Testträger integrierten KIFLEX-Gesamtsystems durchgeführt.

6. Fortschreibung des Verwertungsplans

Erfindungen und Schutzrechtsanmeldungen

Keine.

Wirtschaftliche Erfolgsaussichten nach Projektende

Die entwickelten Verfahren zur Echtzeit-Objektdetektion mittels neuronaler Netze auf Basis von auf Rohdatenebene fusionierten Kamera- und Lidardaten versprechen hohe wirtschaftliche Erfolgspotentiale.

Das FOKUS wird die Ergebnisse des Projekts in verschiedenen Bereichen nutzen. Vor allem wird das gewonnene Know-how bei der Akquisition von neuen Industrie- und Forschungsprojekten verwendet werden.

Insbesondere mit der Weiterentwicklung der Umfelderkennung und Fusion und dem späteren Einsatz ergeben sich neue Geschäftsfelder durch die Zusammenarbeit mit potentiellen Partnern im Bereich Perzeption und Sensor-Fusion.

Die gewonnenen Erkenntnisse werden zur Akquise weiterer Forschungsprojekte aus öffentlicher Förderung und in Direktbeauftragung der Automobilindustrie genutzt.

Wissenschaftliche und technische Erfolgsaussichten

Das Fraunhofer Institut FOKUS nutzt die im Projekt erzielten Ergebnisse zur Stärkung des wissenschaftlichen Profils im Bereich der Multi-Sensor Perzeption. Speziell die KI-basierte Echtzeit-Objektdetektion in Lidardaten haben hohe wissenschaftliche Potentiale.

Die erzielten Ergebnisse dienen zum Nachweis der wissenschaftlichen Kompetenz des FOKUS. Die Ergebnisse fließen in die Entwicklung der FOKUS Datentoolchain FLLT ein und werden diese verbessern.

Das FOKUS ist in Kooperation mit der TU Berlin in der Lehre aktiv und wird die gewonnen Ergebnisse zur Verbesserung der Lehrinhalte und ggf. in Masterarbeiten nutzen.

Im Projekt konnte das FOKUS seine wissenschaftlichen Kompetenzen im Bereich Perzeption und Fusion weiter ausbauen. Durch die Entwicklung des KIFLEX-Softwareframeworks wurden Kompetenzen im Bereich Künstliche Intelligenz und Grundlagen des Automatischen Fahrens aufgebaut.

Wissenschaftliche und wirtschaftliche Anschlussfähigkeit

Das Fraunhofer Institut FOKUS wird die im Projekt gewonnenen Erkenntnisse nutzen, um den Stand der Technik speziell im Bereich der kombinierten Perzeption von Kamera und Lidar auszubauen und neue Erkenntnisse zum wissenschaftlichen Stand der Technik beizutragen.

7. Anhang

KIFLEX - Deep learning technical report

AP 2.5.13/2.5.14/3.3.10 Training und Validierung der KI-Algorithmen

K. Sharma

Problem Statement

Develop perception algorithms based on camera and lidar sensors for the reconfigurable hardware platform for AI-based sensor data processing for autonomous driving. For a given camera image perform object detection (bounding box + object class), similarly for a given point cloud from lidar detect object class and bounding box.

Related Work - Literature Survey

Object detection is a computer vision problem of localizing and classifying the objects in a scene. Beyond 2D scene understanding, 3D object detection is crucial and indispensable for many real-world applications, such as autonomous driving and domestic robots. Fundamental to its core is the ability to perceive the objects present in the 3D scene from its sensor (Cameras, 3D sensors) to plan its motion safely in real-time.

Over the past few years we have seen a plethora of methods that tackle the problem of 3D object detection by exploiting different data sources.

Image based 2D object detection: Starting with the seminal work of Girshick et al. [34] it was established that convolutional neural network (CNN) architectures are state of the art for detection in images. The series of papers that followed [35, 36] advocate a two-stage approach to this problem, where in the first stage a region proposal network (RPN) suggests candidate proposals. Cropped and resized versions of these proposals are then classified by a second stage network. Different from the two-stage detection pipeline that first predicts proposals and then refines them, single-stage detectors directly predict the final detections. YOLO [38] and SSD [37] are the most representative works with real-time speed. YOLO [38] divides the image into sparse grids and makes multi-class and multi-scale predictions per grid cell. SSD [37] additionally uses predefined object templates (or anchors) to handle the large variance in object size and shape. Recently RetinaNet [39] shows that single-stage detector can outperform two-stage detector if class imbalance problem during training is resolved properly.

Image based 3D object detection: Early approaches to 3D object detection focus on camera based solutions with monocular or stereo images [2, 3]. However, they suffer from the inherent difficulties of estimating depth from images and as a result perform poorly in 3D localization.
[4, 5] leveraged the geometry constraints between 3D and 2D bounding box to recover the 3D object pose. [6, 7, 8] exploited the similarity between 3D objects and the CAD models. Chen et al. [9] formulated the 3D geometric information of objects as an energy function to score the pre-defined 3D boxes. These works can only generate coarse 3D detection results due to the lack of depth information and can be substantially affected by appearance variations.

Lidar based 3D object detection: Object detection in point cloud is intrinsically 3 dimensional problem. These methods are generally based on using directly 3D points or different representation of point cloud such as voxel, bird-eye view.

1. Vote3D [28] uses sliding window on sparse volumes in a 3D voxel grid to detect objects. Hand-crafted geometry features are extracted on each volume and fed into an SVM classifier. Vote3Deep [29] also uses the voxel representation of point clouds, but extracts features for each volume with 3D CNN. Song et al. [15] and Zhou et al. VoxelNet[16] grouped the points into voxels and used 3D CNN to learn the features of voxels to generate 3D boxes. SECOND [30] offered a series of improvements to VoxelNet resulting in stronger performance and a much improved inference speed. However, they were unable to remove the expensive 3D convolutional layers. The main issue with voxel representations is efficiency, as the 3D voxel grid usually has high dimensionality and 3D CNN is both memory and computation inefficient.

1. In contrast, VeloFCN [20] projects the 3D point cloud to front-view and gets a 2D depth map. Vehicles are then detected by applying a 2D CNN on the depth map. 3DFCN [31] exploits a bird's eye view representation of the LIDAR and applies a 3D fully convolutional network. PIXOR [12] conducts a single-stage, proposal-free detection over a height-encoded bird's eye view representation.

- Instead of using derived representations, some methods directly use point cloud data. Qi et al. [19] presented the PointNet architecture to directly learn point features from raw point clouds, which greatly increases the speed and accuracy of point cloud classification and segmentation. The follow-up works [20, 27] further improve the extracted feature quality by considering the local structures in point clouds. PointRCNN [32] directly generates 3D proposals from raw point cloud in a bottom up manner, and refine proposals to obtain the final detection results. Recently developed method PointPillars [33] utilizes PointNet [19] to learn a representation of point clouds organized in vertical columns (pillars) and directly predicts 3D objects without generating proposals.

Joint Image - Lidar based 3D object detection: Over the past few years, many techniques have explored both cameras and 3D sensors jointly to perform 3D reasoning. MV3D [10] also uses the projection representation. It combines CNN features extracted from multiple views (front view, bird's eye view as well as camera view) to do 3D object detection. It generates 3D proposals from LiDAR features, and refines the detections with ROI feature fusion from LiDAR and image feature maps. AVOD [11] further extends ROI feature fusion to the proposal generation stage to improve the object proposal quality. However, ROI feature fusion happens only at high-level feature maps. ContFuse [14] uses continuous convolution [30] to fuse multi-scale convolutional feature maps from each sensor, where the correspondence between image and BEV space is achieved through projection of the LiDAR points. F-Pointnet, PointFusion [17, 18] utilized mature 2D detectors to generate 2D proposals from images and reduced the size of 3D points in each cropped image regions. PointNet [19, 20] is then used to learn the point cloud features for 3D box estimation. But the 2D image- based proposal generation might fail on some challenging cases that could only be well observed from 3D space. Such failures could not be recovered by the 3D box estimation step. In this framework the overall performance is bounded by each stage which is still using single sensor

Summary: While the recently developed 2D detection algorithms are capable of handling large variations of viewpoints and background clutter in images, the detection of 3D objects with point clouds still faces great challenges from the irregular data format and large search space of 6 Degrees-of-Freedom (DoF) of 3D objects.

The gap between the two remains large on standard benchmarks such as the KITTI Object Detection Benchmark [1] where 2D car detectors have achieved over 90% Average Precision (AP), whereas the top scoring 3D car detector on the same scenes only achieves 70% AP. The reason for such a gap stems from the difficulty induced by adding a third dimension to the estimation problem, the low resolution of 3D input data, and the deterioration of its quality as a function of distance. Furthermore, unlike 2D object detection, the 3D object detection task requires estimating oriented bounding boxes. However, each sensor has its challenge: cameras have difficulty capturing fine-grained 3D information, while LiDAR provides very sparse observations at long range.

Camera Image-based Network

Input

For this network, we take an RGB camera image as the input, and we normalize it in the range [-1,1] before passing it to the network architecture.

Network architecture

The deep learning network is based on a one-stage approach. Figure 1. shows the overall meta-architecture, which takes the normalized camera RGB image as an input into a CNN module that is used for feature extraction (base network). Well established network architectures, such as InceptionNet-v2 is employed as the base network. From this base network, the last feature map, as

well as intermediate ones, are taken as inputs into the subsequent feature network. The feature network aggregates the feature maps, adds further convolution layers or builds a feature pyramid to prepare the features for the subsequent prediction network. The prediction network adds layers for the classifier and the regression bounding boxes of the objects, followed by a non-maximal suppression step.

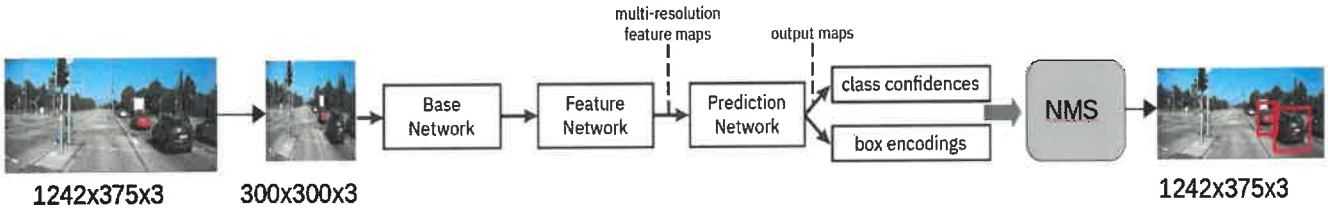


Fig.1 Meta architecture

The one-stage approaches are based on the idea of predicting object classes for a set of anchor boxes and regress refinements relative to these anchors. For this purpose, a grid of size H_grid × W_grid is laid over the input image as can be seen in Figure 2. For each cell of this grid, T anchor boxes (template boxes) of different sizes are created. These anchor boxes cover the whole image and act as dense proposal boxes.

Figure 2: For one-stage architectures, a grid is placed on top of the input image. Multiple anchor boxes with different dimensions are generated for each cell of the grid

Our SSD architecture uses Inception V2 to extract features from the input image. The SSD approach takes different object scales into account and utilizes the importance of regionality for box detection by using convolutional layers instead of fully connected layers. The SSD architecture accounts for varying object sizes by computing feature maps of different shapes and makes use of the fact that smaller structures are handled in earlier layers of the CNN hierarchy whereas larger features emerge in the last layers. In contrast to YOLO, multiple grids of different size are created for the input images. In the original approach, six such grids were generated. For an input image of size 300 × 300, these grids have the following shapes: 38×38, 19×19, 10×10, 5×5, 3×3, and 1×1. For each of those grids, anchor boxes with varying size and aspect ratio are generated. For the first grid and the two last grids, four anchors per cell and for all other grids, six anchors per cell are generated. This leads to 8732 anchor boxes in total. Each of these grids corresponds to two output maps of the CNN that have the same height and width as the grid. The number of channels of these maps corresponds to the number of classes and the regression output. One of the maps per grid handles the classification and thus has a shape of H grid × W grid × T (C + 1) where C denotes the number of classes and the additional channel corresponds to the background class. The other map is used as a regression layer with four channels as output for box position and size relative to the position and size of the anchor box.

Post processing

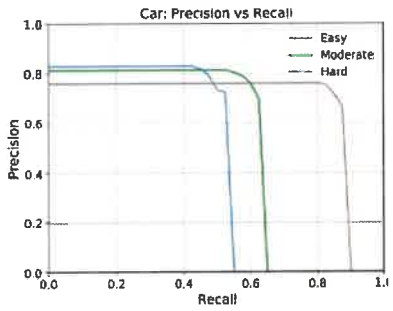
The network predicts a large number of overlapping object boxes, which are merged using Non-maximum suppression algorithm in order to obtain the final predictions.

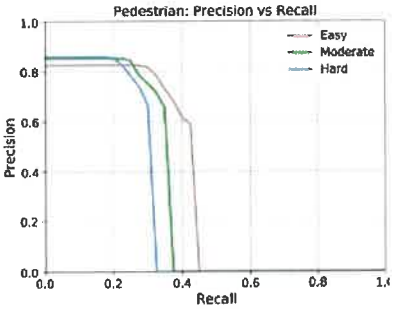
Performance Analysis

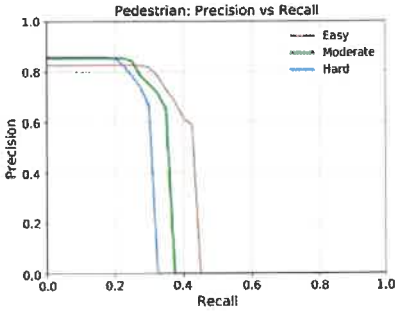
Dataset: For training and evaluation, we have used KITTI dataset. From which we have used 5236 images for training and 2245 images for validation.

Performance: The performance of the network is as shown in the table below showing average precision and figure below shows the precision recall curves.

Method	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
ImageNet-1242x345	48.14	80.70	73.51	64.88	48.88	40.18	35.57	37.85	30.74	29.03
ImageNet-300x300	39.21	66.39	51.09	43.92	34.51	30.01	26.21	38.36	36.54	34.43







Compression Evaluation:

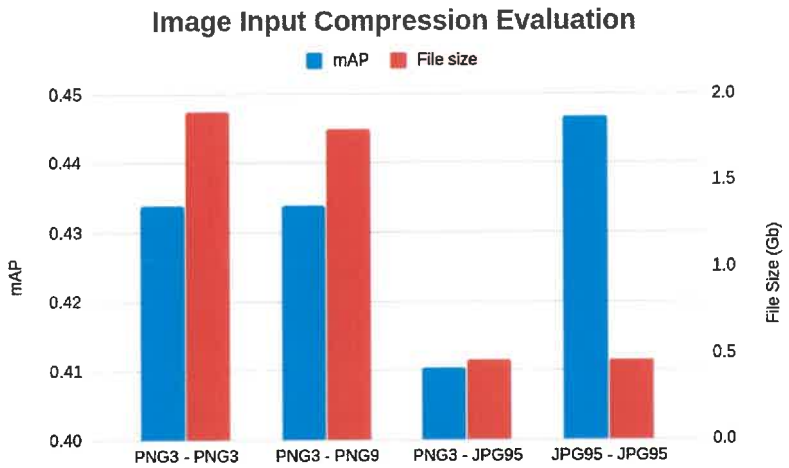
The current image network is trained on two variants of images, first on original PNG3 Quality and second on converted and compressed JPG 95 quality from training set. These networks are evaluated on PNG3, PNG9 and JPG95 quality images from validation set

Following are different experiments for image based network:

- Training – PNG 3 | Evaluation – PNG 3

- Training – PNG 3 | Evaluation – PNG 9
- Training – PNG 3 | Evaluation – JPG 95
- Training – JPG 95 | Evaluation – JPG 95

	Val TF-record size	mAP	Car	Pedestrian	Cyclist
PNG3 - PNG3	1.9 Gb	0.43384	0.67571	0.36826	0.25755
PNG3 - PNG9	1.8 Gb	0.43384	0.67571	0.36826	0.25755
PNG3 - JPG95	464.9 Mb	0.41037	0.65379	0.36594	0.21139
JPG95 - JPG95	464.9 Mb	0.44677	0.69486	0.39007	0.25537



Remarks: The same model is trained on two different image inputs with the same training configuration. As seen from the validation TF-record file sizes, JPG95 compresses the file size to 75% as compared to the original PNG3 format. When the JPG 95 validation set is evaluated using a model trained on PNG3, the performance decreases slightly. But if the same validation set is evaluated using a JPG 95 training set model, the performance is better than other variants. However this is not the actual representation for comparison, as the training is carried out using the same configuration, and different inputs require different training configuration (Hyper parameter tuning). Nonetheless, this evaluation shows, even if the image file size is compressed to 75% of original size, the effect on the performance is not that much severe.

Lidar occupancy grid-based network

Input representation

Point clouds can be represented in many different ways. The raw point cloud is usually given as an unordered set of points (cartesian coordinates and reflectivity). Depending on the sen- sor, the point cloud might be sorted by the azimuthal angle or given in spherical coordinates. In order to use deep learning methods on point clouds, a suitable representation is needed.

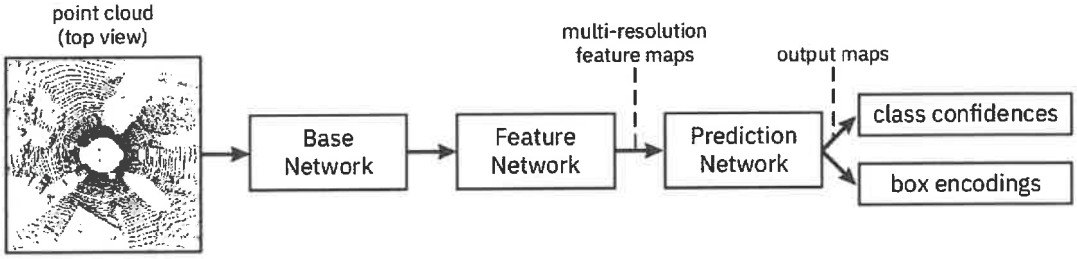
Fig.3 Camera image and corresponding occupancy grid image

These top view representations are created with different cell sizes and areas of interest around the ego-vehicle. Experiments have been performed on grids that capture an area of 150 m × 150 m with cell resolutions of 0.15 m/px . resulting in an occupancy grid image of size 1000 x 1000 px. Each grid cell encodes the minimum and maximum height as well as the maximum reflectivity of the contained points (z min , z max , r max). The top view representations are encoded in 8 bit RGB images (R: z min , G: z max , B: r max). The height values are relative to the height of the laser scanner and limited to the interval [−5 m, 3 m] such that −5 m ∼ 0 and 3 m ∼ 255 so that a resolution of 8m 255 ≈ 3.14 cm is achieved. The reflectivity values are given in the interval [0, 1] and scaled to [0, 255]. Since the input of the network should be normalized, each top view image channel C ∈ {R, G, B} is normalized to the interval [−1, 1] before the image is fed into the neural network:

Network architecture

The network architecture of lidar occupancy grid-based network is same camera image based network mentioned above. For the base network, we similarly use Inception V2 network. A feature pyramid is added to the Inception V2 base network by using the last layer ("mixed 5c") as first feature map. This feature map is then up-sampled and combined with the "mixed 4c" unit that results in the second feature map. This feature map is up-sampled again and added to the "mixed 3c" unit that results in a

third feature map. For an input occupancy grid image of size 1000 × 1000, the feature maps have the following shapes: 32×32, 64×64, 128×128.



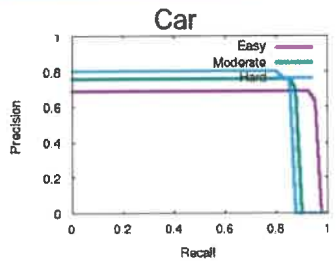
Post processing:

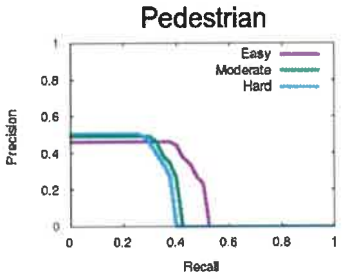
Once the encoded boxes are predicted, they are decoded and NMS is applied to merge overlapping boxes in order to obtain the final detections. To increase the accuracy at runtime, a modified version of NMS is proposed that takes the object orientation into account without the need to determine the IoU of the oriented boxes. Using the separating axis theorem (SAT), a collision check of the detected objects is performed instead of the computation of the IoU. That way, the processing is faster compared to determining the IoU of oriented boxes but more accurate than using an overlap threshold of non-oriented boxes

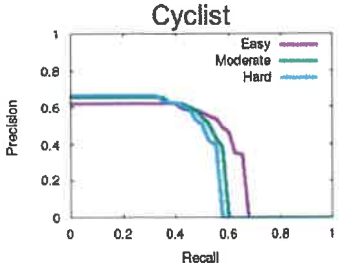
Performance Analysis

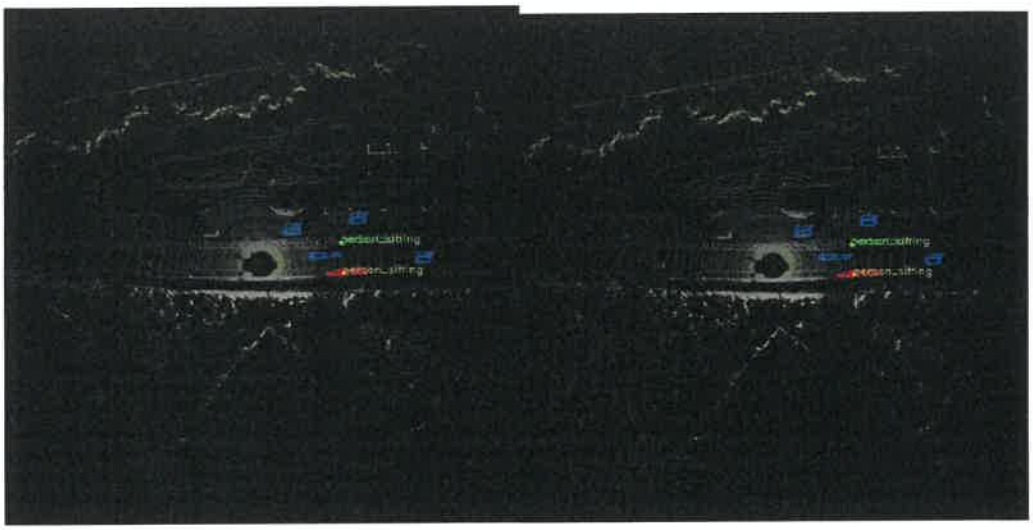
Dataset: For training and evaluation, we have used KITTI dataset. From which we have used 5236 velodyne point clouds for training and 2245 point clouds for validation.

Method	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
LidarNet-1000 x 1000	0.39	0.84	0.73	0.66	0.25	0.21	0.19	0.25	0.23	0.22







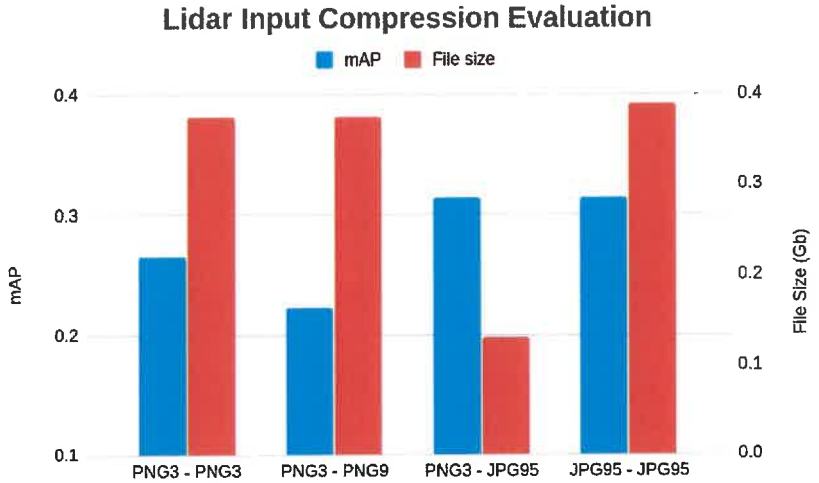


Compression Evaluation: The current lidar network (OccgridNet) is trained on two variants of images created by processing point cloud to occupancy grid, The occupancy grid after creation is stored in two formats, first on original PNG3 Quality and second on converted and compressed JPG 95 quality from training set. These networks are evaluated on PNG3, PNG9 and JPG95 quality images from the validation set.

Following are different experiments for Lidar based network:

- Training – PNG 3 | Evaluation – PNG 3
- Training – PNG 3 | Evaluation – PNG 9
- Training – PNG 3 | Evaluation – JPG 95
- Training – JPG 95 | Evaluation – JPG 95

	Val TF-record size	mAP	Car	Pedestrian	Cyclist
PNG3 - PNG3	265.7 Mb	0.37516	0.64941	0.23595	0.24011
PNG3 - PNG9	222.8 Mb	0.37516	0.64941	0.23595	0.24011
PNG3 - JPG95	314.1 Mb	0.13064	0.28288	0.09212	0.01692
JPG95 - JPG95	314.1 Mb	0.38903	0.72900	0.21074	0.22736



Remarks: The same training configuration is used for training the models. As opposed to our expectation, the validation TF-record file sizes increases while using JPG95 compression as compared to PNG3 format. When JPG 95 validation set is evaluated using a model trained on PNG3, the performance decreases tremendously by more than 50%. But if the same validation set is evaluated using a JPG 95 training set model, the performance is better than other variants. However this is not the actual representation for comparison, as the training is carried out using the same configuration, and different inputs requires different training configuration (Hyper parameter tuning). None the less, this evaluation shows, the images can be stored in JPG 95 and a model trained on the same format of images should be used.

References

- A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012, pp. 3354–3361.
- Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In CVPR, 2016
- Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In NIPS, 2015.
- Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Košecká. 3d bounding box estimation using deep learning and geometry. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 5632–5640.IEEE, 2017.
- Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. 2019.
- Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR), pages 2040–2049, 2017
- Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 3936–3943. IEEE, 2014.
- Roozbeh Mottaghi, Yu Xiang, and Silvio Savarese. A coarse-to-fine model for 3d pose estimation and sub-category recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 418–426, 2015.
- Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2147–2156, 2016.
- Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1907–1915, 2017.
- Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh and Steven Lake Waslander. Joint 3d proposal generation and object detection from view aggregation. CoRR, abs/1712.02294, 2017.

- Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7652–7660, 2018.
- Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 641–656, 2018.
- Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In Conference on Robot Learning, pages 146–155, 2018.
- Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 808–816, 2016.
- Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CoRR, abs/1711.06396, 2017.
- Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. CoRR, abs/1711.08488, 2017.
- Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Point fusion: Deep sensor fusion for 3d bounding box estimation. CoRR, abs/1711.10871, 2017.
- Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1(2):4, 2017.
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems, pages 5099–5108, 2017.
- Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 922–928. IEEE, 2015.
- Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 3, 2017.
- Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 190–198. IEEE, 2017
- Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5648–5656, 2016.
- Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In Proceedings of the IEEE international conference on computer vision, pages 945–953, 2015.
- Hao Su, Fan Wang, Eric Yi, and Leonidas J Guibas. 3d assisted feature synthesis for novel views of an object. In Proceedings of the IEEE International Conference on Computer Vision, pages 2677–2685, 2015.
- Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2626–2635, 2018.
- D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In Robotics: Science and Systems, 2015.
- M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 1355–1361. IEEE, 2017.
- Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely embedded convolutional detection. Sensors, 18(10), 2018
- Li, B.: 3d fully convolutional network for vehicle detection in point cloud. In: IROS (2017)
- Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In CVPR, 2019
- Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. CVPR, 2019.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014
- S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
- K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-´ CNN. In ICCV, 2017.

- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European Conference on Computer Vision, pages 21–37. Springer, 2016
- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 779–788, 2016
- T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, 2017