**1. Hinführung**

* + Aufgabenstellung
  + Herausforderung
  + Nutzen der Anlage

**2. Stand der Technik**

* + Kollaboration
  + Roboterarbeiten mit hoher Präzision
  + Visuelle Erkennung von Objekten und Merkmalen
  + **Kameratechnik**
    - 2D
    - 3D -> Unterschiede und Gegenüberstellung der Verfahren (Laufzeit, Triangulation)
  + **Bildverarbeitungsmethoden**
    - Klassisch -> Filter, Template Matching etc.
      * Gut für immer gleiche Objekte und Merkmale, die sich wenig unterscheiden
    - Machine Learning
      * Lernen eines tieferen Verständnisses
      * Erkennung von Klassen, die jedoch unterschiedlich aussehen können -> verbogene Haken
  + **Bewegung des Roboters**
    - Positionsbasierte Bewegungen -> feste Koordinaten für repetitive Aufgaben
    - Regelungasansätze
      * Sinnvoll für variierende Umgebungen
      * Visual Servoing
        + Position Based
        + Image Based
      * Vergleich von Verfahren -> Vorteile, Implementierungsaufwand, Herausforderungen
  + **Hardware**
    - Verwendeter Roboter
    - Gegenüberstellung von Kameratechniken

**3. Bildverarbeitung**

* **Einleitung**
  + Welche Eigenschaften / Merkmale haben die Haken?
  + 4 Modelle
    - Gemeinsamkeiten/Unterschiede
    - Welche Herausforderungen ergeben sich, mit einer Methode möglichst alle 4 Modelle zu erschlagen?
  + Welche Information sind wichtig für die Lösung der Aufgabe?
    - In Zusammenhang mit verschiedenen Ansätzen der Steuerung/Regelung des Roboters
  + Welche Methoden stehen zur Auswahl

* **Gestaltung der Umgebung**
  + Welche Art Hintergrund ist sinnvoll?
  + Wie sollte die Beleuchtung gestaltet sein?
    - Simulationen von Lichtstrahlen
    - Vergleich von Lichtquellen und Auswirkung
  + Finden sinnvoller Beleuchtungen im Kontext der Anwendung
  + Regelung der Beleuchtungshelligkeit

* **Klassische Bildverarbeitung ohne ML**
  + Einführung -> Was versteht man unter klassischer Bildverarbeitung -> Anwendungsbeispiele zur Einordnung
  + Sinnvolles Merkmal ist definitiv die Form und die Kanten
    - Farbe eher sinnlos -> schwarz und glänzt
  + Template Matching
    - Einfache Herangehensweise, findet jedoch nur sehr ähnliche Haken
    - Bietet dann nur eine Information, wo sich der Haken allgemein im Bild befindet
      * Keine extrahierte Information über die Form, die Spitze
  + Finden von Haken über Keypoints/Deskriptoren - vgl. SIFT, HOG, ORB
    - Problematisch weil schwarz und glänzend -> schlechte Eigenschaften für Keypoints
      * Keypoints brauchen Strukturen, Kontraste -> glatte Oberflächen schwierig
  + Weiteres Problem -> Handling von Instanzen
    - Es gibt mehrere Haken, die über die Leiste oben verbunden sind
    - Kantenerkennung (Sobel, Canny) finden zwar die Kanten, betrachten aber alles als EIN Objekt
      * Es gibt keine Trennung von Instanzen
    - Man könnte mit festen Bereichen arbeiten
      * Setzt aber voraus…
        + Dass man die Leisten vorher schon kennt
        + Nur bestimmte Leisten vorkommen dürfen
        + Kameraparameter und Abstände immer gleich sein müssen
      * Schlechte Voraussetzungen für Kollaboration - generell schlecht, wenn sich Zielobjekt ändern darf und trotzdem verarbeitet werden soll
  + Tieferes Verständnis der Haken und Unterscheidung von Instanzen benötigt

* **Bildverarbeitung mit Machine Learning**
  + Einführung -> Was ist der Vorteil von Machine Learning -> Aufbau von Verständnis
    - Erklärung von verschiedenen Ansätzen
      * Klassifizierung vs. Regression
      * Klassisches NN -> Neuronen
      * Faltungsnetzwerke -> lernen von Filtern
      * Kombination beider Techniken -> Klassifizierung und Regression möglich
      * Segmentierung -> Faltungsnetze, U-Netze oder Pyramiden-Netze
      * Komplexere Architekturen -> Faster-RCNN und Yolo nennen
  + **Segmentierung** und Instanzsegmentierung in Verbindung mit der vorliegenden Aufgabe
    - Warum ist Segmentierung sinnvoll -> pixelweise Zuordnung -> erlaubt spätere Berücksichtigung der Hakenform
    - Instanzsegmentierung -> Ein Bild enthält immer mehrere Haken
      * Später muss aber für das Einfädeln ein Haken gewählt werden, mit dem dann der gesamte Einhänge-Prozess durchgeführt wird
  + **Datenset**
    - Was ist bei der Erstellung wichtig, zu beachten?
    - Wie viele Bilder, wie viele Hakeninstanzen?
    - Möglichkeiten zu Erhöhung der Varianz -> Augmentation, Spiegelung, Rotation, etc.
    - Wie wurde gelabelt?
      * Masken -> wie sollten die Masken sinnvollerweise gezogen werden -> beachte: später interessieren uns eigentlich nur noch Punkte
    - Verteilung von Hakenmodellen und Anzahl Samples

* + **Faster-R-CNN**
    - Erster Ansatz
    - Beschreibung der Architektur -> Warum gerade sinnvoll für unsere Aufgabe?
    - Darstellung Input -> Output
      * Netz bekommt Bilder mit mehreren Haken
      * Netz liefert Output mit mehreren Bboxes, Klassifizierung und Confidence-Werten
    - Verwendetes Pytorch-Modell -> vortrainiert ResNet50 -> bekannte Benchmarkwerte
    - Trainingsverlauf
      * Optimizer -> Welcher Optimizer wird verwendet, was zeichnet diesen aus?
      * Loss-Funktionen -> Loss setzt sich aus mehreren Loss-Fkt zusammen -> warum und welche sind sinnvoll?
      * Parameter -> Learning Rate, Epochen
      * Lossgrafiken etc
        + Welche Lossarten sind besonders hoch und warum?
        + Overfitting auf den Trainingsdaten
      * Dauer Training
    - Inferenz
      * Welche Genauigkeit auf Testdaten konnte erreicht werden?
    - Non-Maximum-Suppression
      * Auswahl der BoundingBoxen am Output des Netzes
      * Wie wird ausgewählt
      * Ablaufdiagramm NMS
    - Laufzeitmessung
      * Single Image Inference Geschwindigkeit auf CPU und GPU
      * Eingebettet in ROS2 -> CPU vs GPU
      * Probleme von zu langsamer Laufzeit -> vor allem für Regelungen des TCP schwierig, da langsames Feedback -> verlangsamt den Prozess extrem

* + **Yolo-Netzwerke**
    - Zweiter Ansatz zur Steigerung der Geschwindigkeit
    - Allgemeine Beschreibung
      * Was ist Ziel von Yolo/Ultralytics -> Einsatzgebiete
      * Überblick über die Modelle -> Modell 8 ist unseres (erstes Modell mit Instanzsegmentierung)
    - Aufbau des Modells
      * Welche Bausteine sind wofür zuständig?
      * Was macht diese Netze so schnell?
      * Kurzer Überblick über Modellgrößen (Nano, Small, Medium)
    - Folgendes wird jeweils für alle drei Modellgrößen verglichen
      * Trainingsverlauf
        + Optimizer -> Welcher Optimizer wird verwendet, was zeichnet diesen aus?
        + Loss-Funktionen -> Loss setzt sich aus mehreren Loss-Fkt zusammen -> warum und welche sind sinnvoll?
        + Parameter -> Learning Rate, Epochen
        + Lossgrafiken etc

Welche Lossarten sind besonders hoch und warum?

Overfitting auf den Trainingsdaten

* + - * + Dauer Training
      * Inferenz
        + Welche Genauigkeit auf Testdaten konnte erreicht werden?
      * Laufzeitmessung
        + Single Image Inference Geschwindigkeit auf CPU und GPU
        + Eingebettet in ROS2 -> CPU vs. GPU

* + **Weiterverarbeitung des Outputs**
    - Information aus Netzoutput beziehen sich momentan nur auf ein einziges Bild auf Pixelebene
    - Nummerierung der Instanzen innerhalb eines Bildes -> von rechts nach links durchnummeriert -> warum sinnvoll, kommt später bei Prozesskapitel
    - Speicherung des Outputs in Dictionary
      * Aufbau Dict für ein Beispielbild

* + **Kleines Fazit zwischen Faster-RCNN vs. Yolo und Ausblick**
    - Warum wird Yolo genommen -> ausreichende Genauigkeit
    - Welche Netzgröße von Yolo wird verwendet
    - Ausblick -> Was fangen wir nun mit dem Netzoutput an
      * Entweder Regelung auf Bildebene
      * Oder Berechnung realer Koordinaten -> braucht dann aber auch Tiefenwerte aus 2D-Bild
      * Momentan liegen nur Informationen auf 2D-Pixelebene vor

**4. Übersetzung der Pixelebene in die reale Welt und Rekonstruktion der Tiefe**

* **Einleitung**
  + Bisheriger Output (Bboxes, Masken, etc) sind nur auf Pixelebene verfügbar
    - Könnte für Image Based Visual Servoing ausreichen
    - Problem: Alle 6 Dimensionen müssen mit Hilfe von Merkmalen regelbar sein
      * Besondere Herausforderung: Regelung der Tiefe Z
      * Kann typischerweise gut mit Hilfe von Größenverhältnissen geregelt werden
        + Bsp QR-Codes -> solange hinfahren, bis sie eine gewisse Größe im Bild haben -> dann kann von einem definierten Abstand Z ausgegangen werden
        + Funktioniert hier jedoch nicht -> wir kennen die Haken vorher nicht
        + Haken haben unterschiedliche Dimensionen und Dicken

Mit Bildern verdeutlichen

* + - * Somit wird jede Art der Regelung zumindest einen Richtwert für die Tiefe/Abstand der Haken zu einem Bezugspunkt erfordern

* **Messung der Tiefe mit Hilfe von Tiefenkameras**
  + Bezug auf Einleitungskapitel
  + In diesem Kapitel werden zwei Methoden getestet -> Triangulation und Time-of-Flight
  + Time-of-Flight -> Intel Realsense D435
    - Was ist Realsense D435 -> Einsatzgebiete (Bin Picking, Aufgaben mit weniger Präzisionsanforderungen)
    - Messung der Tiefe über Laufzeitmessung -> einfaches Prinzip -> Formeln
    - Test mit verschiedenen Abständen -> Ergebnis -> Haken in Tiefenkarte gar nicht sichtbar
  + Stereo Triangulation -> Roboception RC\_visard 65
    - Was ist Roboception RC\_visard65

Messprinzip erläutern mit Formeln -> werden später sowieso noch einmal benötigt beim eigenen Ansatz

* + - Gegenüberstellung Triangulation zu Time-of-Flight in Genauigkeit und Eigenschaften
    - Tests Roboception Kamera aus verschiedenen Abständen -> kein gutes Ergebnis der Tiefenkarten
      * Eigenartige Verzerrung der Tiefenwerte in Tiefenkarte -> deutliche Schwankung die definitiv außerhalb des möglichen liegen von Haken zu Haken
      * Vermutlich durch schwarze Oberfläche und Spiegelungen
        + Vllt gibt’s hier noch eine Quelle, die das untermauern kann

* **Tiefenrekonstruktion über Stereo-Triangulation**
  + Neuer Ansatz -> anstatt Keypoints auf klassische Weise zu finden, findet das NN die relevanten Punkte
    - Betrachtung eines Hakens aus zwei verschiedenen Perspektiven
    - Finden der Punkte Tip und Lowpoint
    - Berechnung von Triangulation
      * Rechnung erläutern -> Bezugnahme Formeln und vektorielle Veranschaulichung aus Kapitel vorher (Roboception)
        + Wo kommen die benötigten Rechengrößen her

Baseline aus Roboterkoordinaten

Disparität aus neuronalem Netz

Intrinsische Kameraparameter vom Objektiv

* + **Berechnung von Triangulationspunkten aus Netz-Output**
    - Neuronales Netz gibt lediglich die Bbox-Koordinaten und die Masken aus
    - Für Triangulation sind vor allem Spitze und Senke des Hakens interessant
      * Müssen zunächst berechnet werden
        + Verschiedene Ansätze

Mittelpunkt der Bbox

einfach zu berechnen

liegt allerdings nicht unbedingt in der tatsächlichen Mitte der Spitze

Untermauen mit Bild mit ungleichmäßiger Maske und Verzerrung der Bbox

Durchschnittswert (Moment) der Masken

vergleichbar mit Schwerpunkt

Erklärung (Formeln)

findet den tatsächlichen Mittelpunkt der Spitze basierend auf der Masken

Verzerrungen in der Maske sind hier weniger problematisch

Finden des Mittelpunktes über KMeans Algorithmus

Bekanntes Machine Learning Verfahren -> kurze Erklärung

Generiert ein Clusterzentrum auf Basis der Verteilung der Pixel, die zur Maske gehören

Liefert das gleiche Ergebnis wie Moment-Berechnung, dauert aber länger

Vergleich der drei Ansätze mit Bildern aus Jupyter Notebook

* + **Vergleich von Triangulations-Verfahren**
    - Horizontale und vertikale Triangulation
      * Änderung der Baseline-Richtung
      * Vergleich beider Verfahren mit Messungen
      * Erklärung des "Versuchsaufbaus" -> Herangehensweise zur Messung der Tiefe
        + Messung 1 -> Triangulation
        + Roboter um 10cm verfahren in z-Richtung
        + Messung 2 -> Triangulation
        + Differenz zwischen Messung 1 und Messung 2 müssen 10cm ergeben
    - Kombinierte Triangulation
      * Bezugnahme auf gefundenes Paper
      * Wie läuft kombinierte Triangulation ab (horizontal + vertikal)
        + Einzelne Berechnung der Tiefe
        + Mittelwert beider Werte
      * Vergleich zwischen horizontaler (1), vertikaler (2) und kombinierter (3) Triangulation
        + Kombiniert am besten
        + Problem -> verlangsamt den Prozess deutlich, da immer doppelt gefahren werden muss für jeden Haken

* + **Einbau in den Prozessablauf -> Entlangfahren und "Scannen" der Haken**
    - Zwei Aufgaben -> Scannen der Haken und globale Zuordnung
    - Bietet später den Vorteil von globaler Adressierung eines Hakens
    - Beschreibung des Prozesses mit Ablauf-Diagramm

* + **Schwingung bei Roboterstopp und Output-Filterung des neuronalen Netzes**
    - Schwingungen durch Roboterstopp
      * Für Triangulation muss Roboter immer wieder gestoppt werden
      * Dabei entstehen Schwingungen des Roboterarms, die sich aufs Kamerabild übertragen
        + Diese übertragen sich wiederum auf die Berechnung der Triangulationspunkte
        + Zeigen von Schwingungsdiagrammen bei unterschiedlicher Geschwindigkeit
        + Beschreibung aus Systemtheorie für Schwingungen

Frequenz -> müsste abhängig von Fahrgeschwindigkeit sein

Verhalten -> asymptotisch

* + - Grundrauschen des Netzoutputs
      * Ergibt sich durch minimal unterschiedliche Werte der Inferenz-Outputs
    - Minimierung beider Schwingungsverhalten mit Filtern
      * EMA-Filter (exponentielle Gewichtung von neuen und alten Werten) -> bringt nicht wirklich viel -> dämpft nur ganz leicht das Grundrauschend es neuronalen Netzes
      * Mittelwertfilter (einfache Implementierung -> arbeitet aber frequenzunabhängig)
      * Tiefpassfilter -> bekannter Filter aus Regelungstechnik
        + Arbeitet frequenzbasiert

Kann also einmal auf Grundrauschen des Netzoutputs eingestellt werden -> unabhängig von Roboterbewegung

Und einmal auf Schwingungsverhalten des Roboterarms, um Schwingungen bei Stoppts auszugleichen

* + - * Vergleich der Ergebnisse aller drei Filter basierend auf gleicher Schwingungsaufzeichung -> Jupyter Notebook
        + Auswahl von sinnvollen Filtern für unsere Anwendung

* + - Filterung von fälschlichen Netzausgaben
      * Für kurze Frames können Hakeninstanzen detektiert werden, die es eigentlich nicht gibt
      * typisches Problem bei neuronalen Netzen -> Fehldetektion
      * Filter basierend auf Frames
        + Eine neue Hakeninstanz muss eine bestimmte Anzahl Frames konstant vorhanden sein, bevor sie tatsächlich zu den validen Instanzen gezählt wird
        + Kurze Beschreibung des Algorithmus

**5. Steuerung bzw.** **Regelung des Roboters**

* **Einleitung**
  + Bezugnahme zu verschiedenen Ansätzen in Stand der Technik
  + Aufgabe beinhaltet verschiedene Schritte, die jeweils unterschiedliche Methoden zu Robotersteuerung bzw. Regelung benötigen
    - Greifen des Bauteils (Positionsbasiert)
    - Heranfahren an Haken (grob positionsbasiert)
    - Genaues Positionieren des Bauteils am Haken (möglicherweise Regelungsbasiert)
    - Einfädeln bzw. Einhängen inkl. Loslassen (Regelungsbasiert)
    - Zurückfahren (Positionsbasiert)
  + Vor allem interessant -> regelungsbasiertes Verfahren für genaue Positionierung und Einfädelung
    - Hier kann nicht mit festen Positionen gearbeitet werden -> Haken unterschiedlich, muss jedes Mal angepasst erfolgen
    - Hierzu Visual Servoing

* **Visual-Servoing-Verfahren im Kontext der Aufgabenstellung**
  + **Image Based**
    - Erklärung des Regelkreises
      * Wie funktioniert die Feedback-Schleife und die Regelung -> Übersetzung von Bildmerkmale auf Roboterbewegung über inverse Bildmatrix
    - Vorteil - translatorische Regelung in x und y-Richtung kann direkt proportional zu den Pixelkoordinaten der Hakenspitze erfolgen
      * Rotation um z auch
    - Probleme
      * Regelungen der Rotation um x und y müsste entsprechend anderer Merkmale erfolgen
      * Kann bei "bekannten" Geometrien zB über Holographie gemacht werden
        + Verzerrungen von Geometrien auf Bildebene geben Auskunft über tatsächliche Lage im Raum
    - Größtes Problem - kein Merkmal bildet die Grundlage für eine Regelung der translatorisch z-Achse genau genug
  + **Position Based**
    - Beschreibung
      * Wie zuvor beschrieben -> durch Stereo Triangulation gibt es bereits reale XYZ-Daten der Haken im Kamerasystem
      * Durch permanente Umrechnung dieser XYZ-Werte auf die aktuelle Roboterstellung lässt sich ein positionsbasiertes Regelungsverfahren realisieren
        + Wenn dies zu ungenau ist, wäre es denkbar, mit Hilfe einer zweiten Kamera in Echtzeit die realen Koordinaten über dauerhafte Triangulation zu berechnen
    - Merkmale zur Regelung des Einfädelvorgangs
      * Orthogonaler Schnitt von Hakengerade und Lochebene als Ziel
        + Mathematische Formeln
        + Übersetzung von Vektorrechnung (Gerade/Ebene) auf Roboterregelung (xyz, roll, pitch, yaw)
    - Test/Validierung des Scan-Prozesses durch Anfahren der Hakenspitze
      * Gescannte Daten für alle Haken zeigen
      * Anfahren jedes Hakens
      * Tabelle mit Abweichungen bzw. Korrekturwerten
      * Gründe für Abweichungen (Offsets) in den Positionen (vor allem in den Tiefenwerten)
        + Diskussion von optischen Verzerrungen während des Scan-Prozesses
        + Durch geringen Abstand zwischen Kamera und Haken (notwendig für Genauigkeit und Auflösung) hat "Sichtkegel" erheblichen Einfluss auf die Darstellung des Hakens im Bild
        + Es wird immer Haken auf rechter Bildhälfte aus Bild 1 mit Haken auf linker Bildhälfte aus Bild 2 verrechnet.
        + Haken sind durch "Sichtkegel" leicht seitlich gezeigt -> einmal von links und einmal von rechts
        + Deshalb unterschiedliche Masken -> unterschiedliche Triangulationspunkte -> Abweichungen im Tiefenwert
    - Einfädelvorgang
      * Weiterverfolgen des Geraden/Ebenen-Ansatzes
      * Berechnung von mehreren Punkten zwischen Hakenspitze und Hakensenke
        + Über Momente von Maskenabschnitten
        + Tiefenwerte für diese Punkte werden zwischen Spitze und Senke interpoliert
      * Gerade schiebt sich dann tangential am Haken entlang weiter
      * Vor allem wichtig für "runde" Haken

**6. Prozessaufbau und Softwarearchitektur**

* Gestell
* Aufbau Kamera/Greifer-Montage
  + Regelung der Beleuchtungshelligkeit
* Ablauf des Prozesses
* Aufbau der ROS2 Infrastruktur
* Teaching mit Nadel
* Arbeitsraum

**7. Auswertung/Fazit**