Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Niki Sillountou

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\,180,30,90,10,180\,]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
5 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [10, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.2, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Michael Khrustalev

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-20,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
5 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [-10, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Lennard Tim Keitel

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,30,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
5 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [20, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Nico Ziegler

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,40,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
5 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Benedict Wandinger

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,30,90,10,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
8 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [-30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Elias Wetzel

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [60,-40,-30,-40,50,60]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
8 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.2, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Daniel Czech

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,30,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
8 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Firas Khedhiri

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-20,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
8 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [-10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Christofer Welser

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,30,90,10,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
6 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Raphael Schmid

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\,180,40,90,-60,180\,]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
6 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [30, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Moritz Höhnel

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,30,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
6 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR3 Roboter die folgende vorgegebene Startstellung:
q = [-30, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.2, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Stefan Leuschner

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-40,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
6 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [10, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Sehun Eom

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\,180,30,90,10,180\,]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
11 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [-10, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Tim Egger

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-20,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
11 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [20, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Maurice Mark

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\,180,30,90,-60,180\,]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
11 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Matthias Maier

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der klassischen DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der klassischen Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-40,30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
11 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [-30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.2, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Kerem Söylemez

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,40,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
15 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [30, -110, 60, -120, 40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Johannes Huber

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-40,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
15 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Daniel Ricardo Pedder

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,40,90,-60,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
15 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [-10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Anna Pinnow

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-20,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
15 Millimeter nach unten entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [10, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Franz Falkenstörfer

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters RoArm-M1, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

q = [180,40,90,-10,180]

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
7 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [30, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.1, 0 , 0 , 0]), a=0.2, v=0.05)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr

Robotik Sommersemester 2025

Prof. Dr. Gerhard Schillhuber

Aufgabenstellung für: Leonard Deiß

Aufgabe 1:

Ziel der Aufgabe:
Sie analysieren die Kinematik eines Roboters myCobot280, indem Sie die Denavit-Hartenberg-Parameter anwenden. Dazu erstellen Sie eigene Fotoaufnahmen des Roboters und kennzeichnen in diesen die Koordinatensysteme gemäß der modifizierten DH-Konvention. Anschließend erstellen Sie die vollständige DH-Tabelle.

Aufgabenbeschreibung:
Fotodokumentation:
Fotografieren Sie den gegebenen Roboter von der Seite und von oben, sodass alle Gelenke und Glieder gut erkennbar sind. Die Bilder sollen klar und möglichst orthogonal aufgenommen sein.
Einzeichnen der Koordinatensysteme:
Zeichnen Sie in beide Bilder die lokalen Koordinatensysteme jedes Gelenks nach der modifizierten Denavit-Hartenberg-Konvention ein. Es reicht, zwei Achsen pro Koordinatensystem einzuzeichnen (z. B. x- und z-Achse). Die dritte Achse ergibt sich aus der Rechtshändigkeit des Koordinatensystems.
DH-Tabelle erstellen:
Bestimmen Sie anhand der Bilder und gegebenen Maße die vier DH-Parameter für jedes Gelenk.

Abgabe:
- Die beiden Fotos mit eingezeichneten Koordinatensystemen.
- Eine schriftliche Erläuterung, wie Sie die Achsen festgelegt haben.
- Die vollständige DH-Tabelle.

Bewertungskriterien:
- Korrekte Anwendung der DH-Regeln
- Saubere und nachvollziehbare Zeichnung der Koordinatensysteme
- Plausible und vollständige DH-Tabelle
- Klarheit der Darstellung und Dokumentation

Aufgabe 2:

Ziel der Aufgabe:
Basierend auf der in der vorherigen Aufgabe erstellten DH-Tabelle, implementieren Sie die Vorwärtskinematik des Roboters. Ziel ist es, die Lage und Orientierung des Endeffektors in Abhängigkeit von den Gelenkwinkeln zu berechnen.

Aufgabenbeschreibung:
Implementierung der Vorwärtskinematik:
Verwenden Sie die in der vorherigen Aufgabe bestimmte DH-Tabelle.
Implementieren Sie die Vorwärtskinematik in Python.
Test mit Gelenkwinkeln:
Gegeben ist der Satz von Gelenkwinkeln

$$q = [\ 60,-40,-30,-40,50,60\ ]$$

Geben Sie die berechnete Transformationsmatrix an, die die Position und Orientierung des Endeffektors im Basis-Koordinatensystem beschreibt.
Fotodokumentation der Stellung:
Bewegen Sie den Roboter in die gegebene Stellung, die den gewählten Gelenkwinkeln entspricht.
Machen Sie ein Foto der Roboterstellung.
Fügen Sie das Foto Ihrer Dokumentation bei, um die berechnete Lage des Endeffektors mit der realen Stellung zu vergleichen.

Abgabe:
- Quellcode-Datei oder Notebook
- Foto der Roboterstellung
- Erläuterung der Vorgehensweise
- Ergebnis der Transformation (T-Matrix und Pose in x,y,z,rx,ry,rz Darstellung)

Bewertungskriterien:
- Korrekte Berechnung der Vorwärtskinematik
- Plausible und vollständige Dokumentation
- Nachvollziehbarer Abgleich von Berechnung und realer Roboterstellung

Aufgabe 3:

Ziel der Aufgabe:
Basierend auf der zuvor berechneten Roboterstellung soll der TCP (Tool Center Point) um eine bestimmte Strecke in z-Richtung verschoben werden – bei gleichbleibender Orientierung des Endeffektors. Zur Ermittlung der neuen Gelenkwinkel ist die Rückwärtskinematik zu implementieren.

Aufgabenbeschreibung:
Zielvorgabe definieren:
Nehmen Sie die zuvor berechnete Pose des Endeffektors.
Verschieben Sie die Position um
7 Millimeter nach oben entlang der z-Achse
des Weltkoordinatensystems.
Die Orientierung des Endeffektors darf sich nicht verändern.
Implementierung der Rückwärtskinematik:
Implementieren Sie die Rückwärtskinematik, um alle möglichen Lösungen für die Gelenkwinkel zu berechnen.
Ergebnis:
Geben Sie die jeweiligen Lösungen als separate Konfigurationen an.
Geben Sie an, welche der Lösungen aus praktischer Sicht (z. B. Erreichbarkeit, Gelenkgrenzen, Kollisionen) sinnvoll oder bevorzugt wäre.
Machen Sie ein Foto der bevorzugten Roboterstellung und fügen Sie das Foto Ihrer Dokumentation bei.

Abgabe:
- Quellcode oder Notebook, das alle Lösungen berechnet und strukturiert ausgibt
- PDF-Dokumentation mit:
- Ursprünglicher TCP-Pose und Zielpose (T-Matrizen)
- Tabelle aller berechneten Gelenkwinkellösungen
- Ggf. grafische oder fotografische Darstellung ausgewählter Konfigurationen
- Kurze Diskussion zur Auswahl bevorzugter Lösung

Bewertungskriterien:
- Vollständige und nachvollziehbare Berechnung aller kinematisch möglichen Lösungen
- Plausibilität der Ergebnisse
- Klare Darstellung und Diskussion der Resultate

Aufgabe 4:

Ziel der Aufgabe:
Ein movel-Befehl, der aus einer vorgegebenen Startstellung ausgeführt wird, soll am realen UR-Roboter getestet und anschließend in Python nachgebildet werden. Dabei werden Soll- und Ist-Werte der Gelenkwinkel und -geschwindigkeiten aufgezeichnet und anschließend mit einer Python-Simulation verglichen. Die Gelenkwinkelgeschwindigkeiten in der Simulation sind mithilfe der Jacobi-Matrix zu berechnen.

Aufgabenbeschreibung:
Startstellung:
Verwenden Sie für einen UR5 Roboter die folgende vorgegebene Startstellung:
q = [-30, -130, 60, -120, -40, 60]

Ausführung des movel-Befehls:
Führen Sie nach Erreichen der Startpose einen movel-Befehl aus, z. B.:
movel(pose_add(get_actual_tcp_pose(),p[0, 0 , -0.05, 0 , 0 , 0]), a=0.1, v=0.04)

Zeichnen Sie während der Bewegung u.a. folgende Daten auf:

- Soll-Gelenkwinkel
- Ist-Gelenkwinkel
- Soll-Gelenkwinkelgeschwindigkeiten
- Ist-Gelenkwinkelgeschwindigkeiten
- Soll-TCP-Pose
- Ist-TCP-Pose
- Soll-TCP-Geschwindigkeiten
- Ist-TCP-Geschwindigkeiten
Speichern Sie die Daten strukturiert (z. B. als CSV mit Zeitstempeln).

Simulation in Python:
Reproduzieren Sie die Bewegung mit einer Python-Funktion:
Verwenden Sie dieselbe Start- und Zielpose wie im realen Versuch.
Interpolieren Sie den movel-Pfad linear im kartesischen Raum.
Berechnen Sie die zugehörigen Gelenkwinkel mithilfe der Rückwärtskinematik.
Bestimmen Sie die Gelenkwinkelgeschwindigkeiten über die Jacobi-Matrix.

Vergleich zwischen realen Daten und Simulation:
Erstellen Sie Diagramme für alle Gelenke, die jeweils vergleichen:
Simulierte Gelenkwinkel vs. Soll-Werte vs. Ist-Werte
Simulierte Gelenkwinkelgeschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Pose vs. Soll-Werte vs. Ist-Werte
Simulierte TCP-Geschwindigkeiten vs. Soll-Werte vs. Ist-Werte
Ermitteln Sie wichtige Trajektorienkennzahlen (Gesamtzeit, Schaltzeitpunkte, Winkel und Geschwindigkeiten zu Schaltzeitpunkten, …)
Ermitteln Sie Fehlerkennzahlen für beide Vergleiche (z. B. RMSE, max. Fehler)

Abgabe:
- Python-Skript oder Notebook
- CSV-Dateien mit real aufgezeichneten Soll- und Ist-Werten
- PDF-Dokumentation mit:
- Start- und Zielpose
- Vergleichsplots der Gelenkverläufe
- Fehleranalyse für:
- Simulation vs. Soll
- Simulation vs. Ist
- Diskussion möglicher Ursachen von Abweichungen

Bewertungskriterien:
- Korrekte Umsetzung der Simulation mit Jacobi-Matrix
- Vollständiger Vergleich Simulation vs. Soll und Ist
- Klare Visualisierung und strukturierte Auswertung
- Plausible Interpretation der Abweichungen

Einreichung: Über Moodle, 10.7.2025, 8:00 Uhr