

Intuitive Steuerung tele-operierter Inspektions- und Wartungsroboter

Konzept zur Benutzerunterstützung durch automatische Objekterkennung und AR-Visualisierung

Dipl.-Inf. **B. Graf**, Fraunhofer IPA, Stuttgart;
Dipl.-Ing. **U. Reiser**, Fraunhofer IPA, Stuttgart

Kurzfassung

Diese Arbeit beschreibt einen Ansatz zur intuitiven Steuerung tele-operierter Inspektions- und Wartungsroboter. Dieser ermöglicht die automatische Erkennung relevanter Objekte mittels eines Time-of-Flight-Sensors, die Augmentierung des zugehörigen Kamerabilds auf der Benutzeroberfläche sowie das automatische Bewegen des Roboterarms zu einem ausgewählten Objekt.

1. Aufgabenstellung

Eine wichtige Aufgabe des Personals industrieller Prozessanlagen besteht darin, regelmäßige Inspektions- und Wartungsaufgaben durchzuführen. Viele dieser Arbeiten sind einfache Routinarbeiten, die durch den Einsatz mobiler Roboter automatisiert werden können, z.B. das Monitoring aktueller Pegelstände oder der Anzeigen von Messgeräten sowie das Auf- oder Zudrehen von Ventilen (Bild 1) [5].

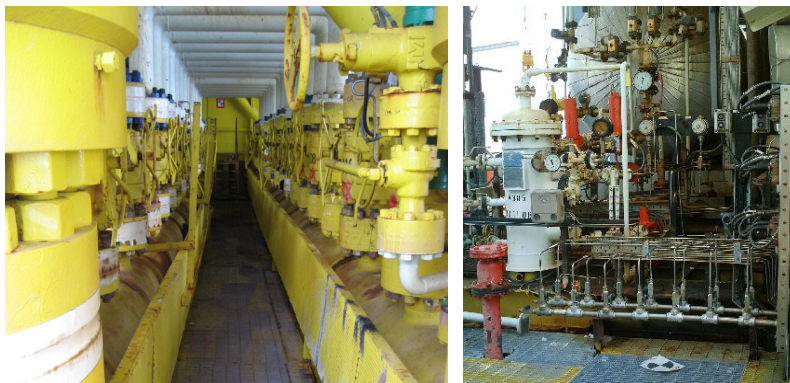


Bild 1 Ventilräder und Anzeigen in einer Prozessanlage

Für den sicheren Betrieb und die Akzeptanz bei den Benutzern ist eine möglichst einfache und intuitive Bedienung des Roboters mit teilautomatischen Sicherheitsfunktionen

notwendig. Dadurch dass die Anzeigen von Messgeräten oder die zu handhabenden Ventile oft eine ähnliche, meist runde Form haben, ist eine automatische Erkennung durch die Sensorik des mobilen Roboters möglich. Durch die teilautonome Führung des Roboterarms können dessen Bedienung durch den Teleoperator vereinfacht und Beschädigungen der Einrichtungsgegenstände, z.B. wenn der Endeffektor des Roboters bei Auf- oder Zudrehen eines Stellrads nicht entlang der korrekten Bahn geführt wird oder bei Kollisionen des Roboterarms mit der Umgebung, vermeiden werden.

2. Lösungsansatz

Die Unterstützung des Teleoperators beinhaltet die automatische Erkennung relevanter Objekte, die Augmentierung des zugehörigen Kamerabilds sowie das automatische Bewegen des Roboterarms zu einem ausgewählten Objekt. In den nachfolgenden Kapiteln werden die einzelnen Komponenten beschrieben, die für das teilautonome Greifen von Stellrädern mit kreisförmigem Handgriff realisiert wurden.

2.1 Automatische Objekterkennung

Die automatische Erkennung kreisförmiger Objekte basiert auf der Verwendung eines Time-of-Flight-Sensors (SwissRanger von Mesa) [7]. Dieser liefert bis zu einem Abstand von ca. 7,5 Metern ein Tiefen- sowie ein Intensitätsbild mit der Auflösung von 176 x 144 Pixeln (Bild 2). Für die eindeutige Identifizierung eines Handrads werden aus dem Tiefenbild die Ebene, in der das Handrad liegt sowie dessen Radius und Drehachse errechnet. Ein großer Vorteil der Verwendung des Time-of-Flight-Sensors besteht darin, dass die bei der Bildverarbeitung gewonnenen Merkmale unmittelbar in 3D-Koordinaten vorliegen. Der Time-of-Flight-Sensor ist mit einer Farbkamera gekoppelt, die das Bild für die graphische Benutzeroberfläche des Teleoperators bereitstellt.



Bild 2 Sensoraufbau, Tiefen- und Intensitätsbild eines Handrads.

Für die Detektion kreisförmiger Objekte im Tiefenbild werden die folgenden Verarbeitungsschritte durchgeführt:

1. Einlesen des Tiefenbilds.
2. Glätten des Tiefenbilds, um den Kontrast für die Erkennung zu erhöhen (Bild 3, rechts).

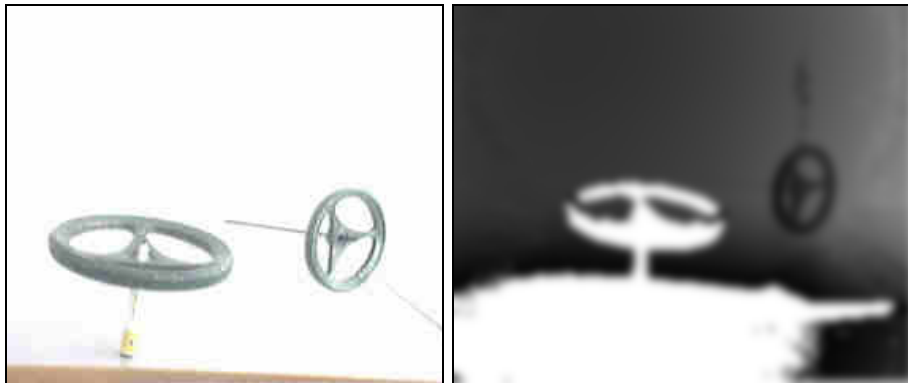


Bild 3 Farb- und Tiefenbild einer Beispielszene und geglättetes Tiefenbild des 3-D-Sensors

3. Kantenextraktion im Tiefenbild (Bild 4, links).
4. Suche nach zu Ellipsen gehörenden Punkten innerhalb des Kantenbilds (2-D-Projektion der gesuchten kreisförmigen Objekte im 3-D-Raum, Bild 4, rechts).



Bild 4 Kantenextraktion und zu Ellipsen gehörenden Bildpunkte

5. Um herauszufinden, welche der identifizierten Ellipsenpunkte zusammengehören, wird Pascals Theorem angewandt, das besagt, dass sechs Punkte $P_1, P_2, P_3, P_4, P_5, P_6$ der selben Ellipse angehören, wenn die Schnittpunkte der Geraden P_1P_2 und P_4P_5 , P_2P_3 und P_5P_6 und P_3P_4 und P_6P_1 auf der selben Gerade liegen (Bild 5, links) [11]. Dieses Theorem wird so lange auf zufällig ausgewählte Ellipsenpunkte angewandt, bis sechs zusammengehörende Punkte identifiziert wurden.

6. Nachdem die zu einer Ellipse gehörenden Punkte im Tiefenbild identifiziert wurden, werden für jede Ellipse anhand der Abstandswerte der Ellipsenpunkte A, B und C die 3-D-Koordinaten der entsprechenden Punkte im Raum errechnet.

7. Aus den Koordinaten (X_A, Y_A, Z_A) , (X_B, Y_B, Z_B) und (X_C, Y_C, Z_C) der drei Referenzpunkte A, B und C wird wie folgt der Normalenvektor für die Kreisebene errechnet (Bild 5, Mitte):

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} X_A & Y_A & Z_A \\ X_B & Y_B & Z_B \\ X_C & Y_C & Z_C \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad (1)$$

8. Der Mittelpunkt O (X_0, Y_0, Z_0) des Kreises wird mittels dreier sich schneidender Ebenen errechnet (Bild 5, links): anhand der Koordinaten (X_A, Y_A, Z_A) and (X_B, Y_B, Z_B) der Punkte A und B kann der in der Mitte zwischen diesen Punkte liegende Punkt E errechnet werden:

$$E \left(\frac{X_A + X_B}{2}, \frac{Y_A + Y_B}{2}, \frac{Z_A + Z_B}{2} \right) \quad (2)$$

Die Ebene S1, zu der die Gerade zwischen A und B orthogonal steht, errechnet sich aus:

$$d_1 = (X_A - X_B) \cdot X + (Y_A - Y_B) \cdot Y + (Z_A - Z_B) \cdot Z \quad (3)$$

Durch Einsetzen von E ergibt sich:

$$d_1 = (X_A - X_B + (Y_A - Y_B) \cdot \frac{Y_A + Y_B}{2} + (Z_A - Z_B) \cdot \frac{Z_A + Z_B}{2}) \quad (4)$$

Punkt F zwischen A und C sowie der Parameter d_2 der entsprechenden Ebene S2 können analog errechnet werden. Die Ebene ABC, in der der gesuchte Kreis liegt ergibt sich dann aus

$$\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} = \begin{pmatrix} X_A - X_B & Y_A - Y_B & Z_A - Z_B \\ X_A - X_C & Y_A - Y_C & Z_A - Z_C \\ a & b & c \end{pmatrix}^{-1} \cdot \begin{pmatrix} d_1 \\ d_2 \\ 1 \end{pmatrix} \quad (5)$$

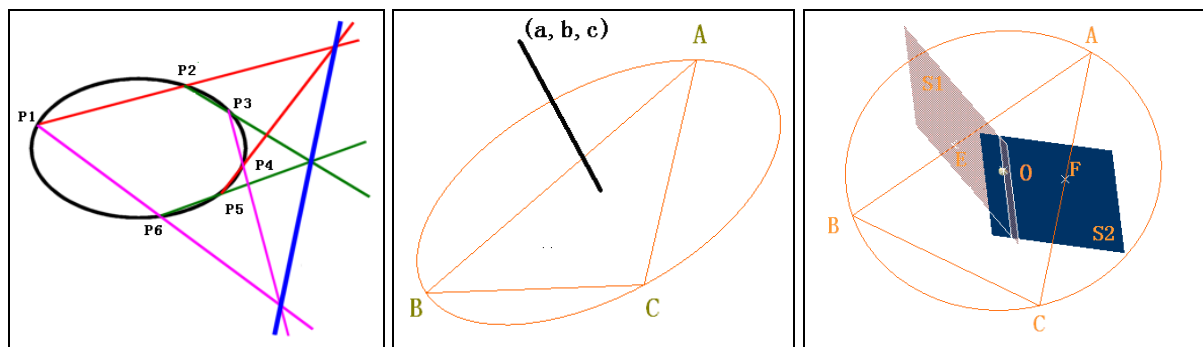


Bild 5 Pascal's Theorem zur Identifikation von Ellipsenpunkten, Kalkulation des Normalenvektors und des Mittelpunkts für den zugehörigen Kreis im 3-D-Raum

9. Der Radius des gesuchten Kreises errechnet sich aus der Distanz zwischen dem Kreismittelpunkt und dem auf dem Kreis liegenden Punkt A:

$$r = \sqrt{(X_o - X_A)^2 + (Y_o - Y_A)^2 + (Z_o - Z_A)^2} \quad (6)$$

2.2 AR-Visualisierung erkannter Objekte

In diesem Schritt wird das augmentierte Kamerabild für den Teleoperator errechnet. Um die Position detektierter Handräder im Bild darzustellen, wird eine Visualisierung von deren Umfang und Drehachse in das Bild eingefügt. Die einzelnen Handräder können über die Benutzerschnittstelle selektiert und entsprechende Bewegungskommandos an den Roboter (z.B. Positionieren vor einer Anzeige, Greifen eines Stellrads, Drehen eines Stellrads) eingegeben werden. Die einzelnen Schritte zur Augmentierung des Kamerabilds werden im Nachfolgenden beschrieben:

10. Einlesen des Videostreams.

11. Anlegen einer virtuellen Kamera (Bild 6, links), die die gleichen Eigenschaften hat wie die verwendete reale Kamera [10].

12. Einfügen des Videostreams als Hintergrundbild für die virtuelle Kamera.

13. Umrechnung der Koordinaten des Kreismittelpunkts (X_o, Y_o, Z_o) aus dem Koordinatensystem des 3-D-Sensors in Kamerakoordinaten (X_o', Y_o', Z_o'):

$$X_o' = X_o + \Delta x \quad (7)$$

$$Y_o' = Y_o + \Delta y \quad (8)$$

$$Z_o' = Z_o + \Delta z \quad (9)$$

$\Delta x, \Delta y, \Delta z$ beschreiben dabei die während der Kalibrierung errechneten Abstände zwischen dem 3-D-Sensor und der Farbkamera.

14. Errechnung der Transformationsmatrix zur Umrechnung der Position und Parameter eines Handrads in Kamerakoordinaten. Ein Punkt (X_w, Y_w, Z_w) im Koordinatensystem des Handrads wird wie folgt in Kamerakoordinaten (X_{Ca}, Y_{Ca}, Z_{Ca}) umgewandelt:

$$\begin{pmatrix} X_{Ca} \\ Y_{Ca} \\ Z_{Ca} \\ 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \quad (10)$$

Der entsprechende Translationsvektor ergibt sich somit als:

$$(T_1, T_2, T_3)^T = (X_o', Y_o', Z_o')^T \quad (11)$$

15. Die Rotationsmatrix wird wie folgt errechnet: Errechne Winkel θ zwischen dem Normalenvektor (a, b, c) und der Z-Achse. Aus dem Normalenvektor (a, b, c) ergibt sich die Rotationsachse:

$$\frac{-b}{\sqrt{a^2+b^2}} X + \frac{a}{\sqrt{a^2+b^2}} Y = 0 \quad (12)$$

Diese Achse ist senkrecht zur Z-Achse und dem Normalenvektor und liegt in der Ebene XOY. Mittels des Winkels θ und der Rotationsachse wird das Quaternion q errechnet, das die Orientierung des Kreises im 3-dimensionalen Raum beschreibt (Bild 6, rechts) [12]:

$$q = (x, y, z, w) \quad (13)$$

mit

$$w = \cos\left(\frac{\theta}{2}\right) \quad (14)$$

$$x = \frac{-b}{\sqrt{a^2+b^2}} \cdot \sin\left(\frac{\theta}{2}\right) \quad (15)$$

$$y = \frac{a}{\sqrt{a^2+b^2}} \cdot \sin\left(\frac{\theta}{2}\right) \quad (16)$$

$$z = 0 \quad (17)$$

Daraus ergibt sich die gesuchte Rotationsachse wie folgt:

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} = \begin{pmatrix} 1-2\cdot(x^2+z^2) & 2\cdot(x\cdot y+w\cdot z) & 2\cdot(x\cdot z-w\cdot y) \\ 2\cdot(x\cdot y-w\cdot z) & 1-2\cdot(x^2+z^2) & 2\cdot(y\cdot z+w\cdot x) \\ 2\cdot(x\cdot z+w\cdot y) & 2\cdot(y\cdot z-w\cdot x) & 1-2\cdot(x^2+y^2) \end{pmatrix} \quad (18)$$

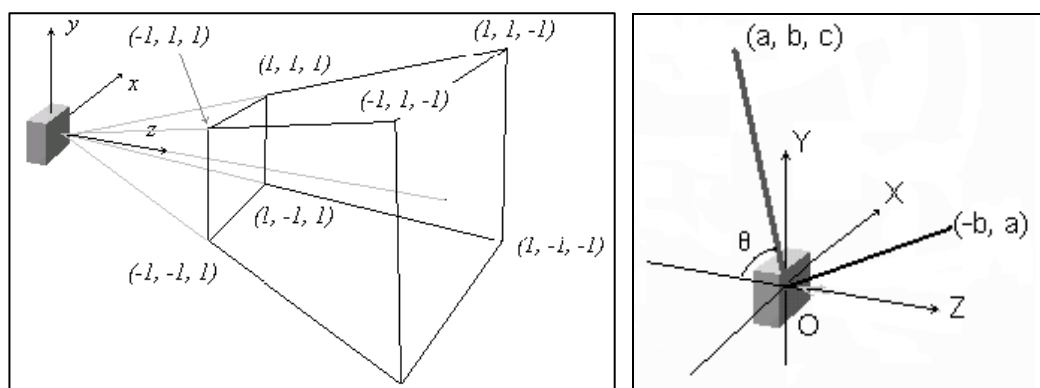


Bild 6 Virtuelle Kamera und Quaternion zur Errechnung der Rotationsmatrix

16. Darstellung der gefundenen Kreise im Kamerakoordinatensystem (Bild 7, rechts)

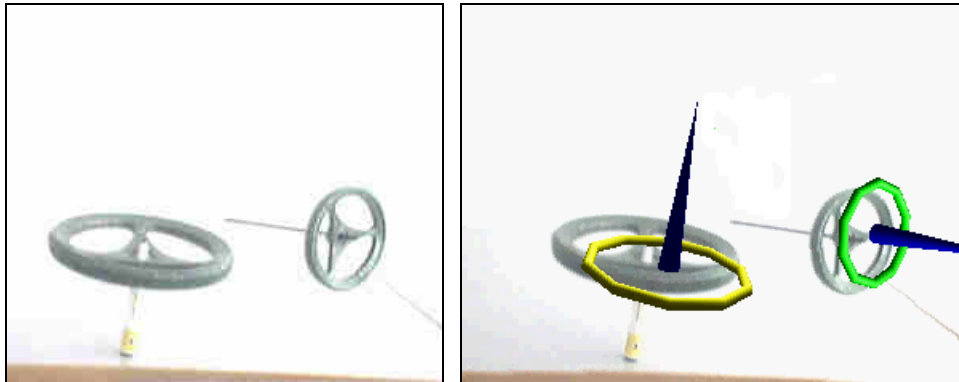


Bild 7 Ursprüngliches und augmentiertes Kamerabild

Um Ungenauigkeiten des 3-D-Sensors auszugleichen, kann der Benutzer auf der graphischen Oberfläche die Eigenschaften der detektierten Handräder manuell anpassen. Danach kann eine Greifposition an einem der Handräder spezifiziert werden, die der Roboter automatisch anfährt.

2.3 Teilautonomes Manipulieren

Nachdem die Greifposition ermittelt wurde, muss der Manipulator die entsprechende Konfiguration einnehmen. Da sich in der Umgebung des Roboters Personen aufhalten können, sind neben einer exakten Kinematik und Gelenkwinkelregelung eine dynamische Kollisionsvermeidung und eine damit verbundene Echtzeit-Überwachung des Arbeitsraumes erforderlich. Die Kollisionsüberwachung schließt sowohl Eigenkollisionen des Roboters als auch Kollisionen mit der Umgebung ein. Industrieroboter sind meist derart konstruiert, dass die einzelnen Gelenke nicht miteinander kollidieren können, so dass eine Eigenkollisionsvermeidung nicht notwendig ist. Insbesondere für mobile Roboter werden jedoch oft Leichtbauarme mit mehr als 6 Freiheitsgraden eingesetzt. Dazu kommt, dass viele mobile Roboter mehrere Manipulatoren besitzen (z.B. für bewegliche Sensorträger, etc.), so dass die Eigenkollisionsüberwachung unabdingbar wird.

Die unterschiedlichen Verfahren der Kollisionsvermeidung können in offline-Methoden wie Bahnplanung und online-Methoden, die während der Manipulatorbewegungen kontinuierlich im Hintergrund ablaufen, gegliedert werden. Für dynamische Umgebungen können prinzipiell nur online-Methoden Sicherheit bezüglich Kollisionsfreiheit mit externen Objekten bieten. Diese stellen allerdings hohe Anforderungen an die Geschwindigkeit der Kollisionserkennungsalgorithmen und der Umweltmodellierung durch die Sensorik.

In dieser Arbeit wird ein Ansatz zur schnellen Kollisionserkennung vorgestellt, der sowohl für die offline- als auch für die online-Kollisionsvermeidung eingesetzt werden kann. Im Rahmen

von Performancemessungen für einen Roboterbau mit 11 Freiheitsgraden konnte für den Algorithmus auf einem handelsüblichen PC (2.0 GHz, 1 GB RAM) eine Zykluszeit von weniger als 1 ms erreicht werden [13]. Da die meisten Robotersteuerungen mit einer Zykluszeit von 6-12 ms arbeiten, ist der Algorithmus für eine Vielzahl von Anwendungen geeignet.

Der Kollisionserkennungsalgorithmus basiert auf dem CULLIDE Paradigma, das aus dem Bereich der Computergraphik stammt [4]. Der Algorithmus besteht aus zwei Hauptschritten:

- Einem groben Distanzcheck aller bewegten Robotergelenke basierend auf deren Geschwindigkeitsvektoren. Alle Teile, für die eine potentielle Kollisionsgefahr besteht, werden in "potential colliding sets" (PCS) zusammengefasst.
- Im zweiten Schritt werden alle PCSs auf tatsächliche Kollisionen untersucht. Dies erfolgt durch Distanzberechnungen aller Teile in jedem PCS, die durch "oriented bounding boxes" (OBBs) modelliert werden.

Für die Eigenkollisionsvermeidung wird der gesamte Roboterbau mit Hilfe von OBBs modelliert. Das zweiphasige Verfahren bietet den Vorteil, dass der Rechenaufwand erheblich reduziert wird, da nur wenige Teile in den PCSs für rechenintensive Distanzberechnungen berücksichtigt werden müssen.

Der Kollisionserkennungsalgorithmus wird im nächsten Schritt mit einem Bahnplanungsverfahren gekoppelt ("single-query probabilistic roadmap path planner with lazy collision checking (SBL)" [1][9]). Der SBL Bahnplaner ermittelt ausgehend von der aktuellen Roboterkonfiguration einen kollisionsfreien Pfad zur Zielkonfiguration. Für die Modellierung der Umgebung werden die vom SwissRanger gelieferten Abstandswerte zur Umgebung in Form einer 3D Punktwolke genutzt (Bild 8).

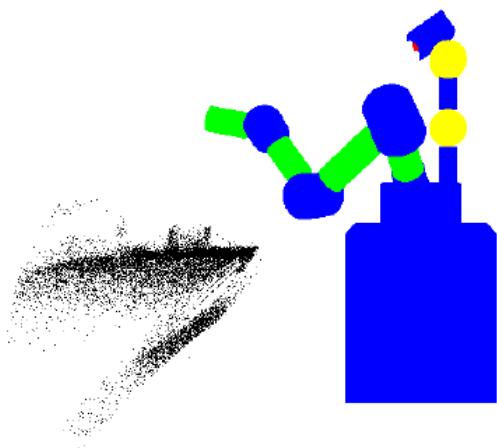


Bild 8 OBB-Modell eines Roboters mit einem 7-DOF-Arm und einem 4-DOF-Sensorkopf sowie und 3D-Punktwolke der Umgebung

Mittels der automatischen Bahnplanung wird eine vorgegebene Position am Handrad sicher und schnell gegriffen, was die Bedienung des Roboterarms für den Teleoperator enorm erleichtert. Für das Drehen von Stellrädern wird nach dem Greifen anhand der Radeigenschaften (Drehachse N, Drehrichtung D, Radius R, Bild 9) eine geeignete Trajektorie errechnet, die den Roboter entlang des Radlaufs führt.

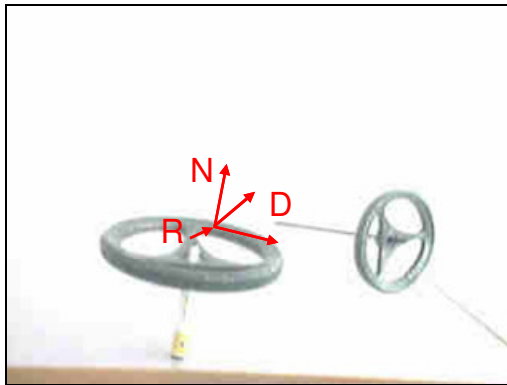


Bild 9 Greifposition und Rotationseigenschaften eines Handrads

3. Referenz

Die diesem Bericht zugrundeliegenden Arbeiten wurden teilweise im Rahmen des Projekts ImRoNet [2] mit Mitteln des Bundesministeriums für Wirtschaft und Technologie unter dem Förderkennzeichen 01MR06002 gefördert.

4. Literatur

- [1] "Motion Planning Toolkit" Retrieved January 23, 2008 from <http://ai.stanford.edu/%7Emitul/mpk>.
- [2] „ImRoNet - Internetbasierte multimediale/multimodale Nutzerschnittstellen zur Teleoperation von Robotern“ (2007), www.imronet.de (8.11.2007).
- [3] Fitzgibbon, Andrew W.; Fisher, R.B.. "A Buyer's Guide to Conic Fitting". In: Proc.5th British Machine Vision Conference, Birmingham,pp. 513-522, 1995.
- [4] Govindaraju, N. K.; Redon, S.; Lin, M. C.; Manocha, D.: CULLIDE, <http://gamma.cs.unc.edu/CULLIDE>.
- [5] Graf, Birgit; Pfeiffer, Kai: "Mobile Robotics for Offshore Automation." In: Penders, Jacques (Ed.) u.a.; Universität Jaume I <Castelló de la Plana, Spanien> u.a.: Robotics for Risky Interventions and Surveillance of the Environment: IARP/EURON International Workshop, January 7th-8th, 2008, Benicàssim, Spain.Castelló de la Plana, Spain, 2008.

- [6] Lourakis, Manolis: "Levenberg-Marquardt nonlinear least squares algorithms in C/C++". Retrieved January 23, 2008 from <http://www.ics.forth.gr/~lourakis/levmar>.
- [7] Reiser, Ulrich; Kubacki, Jens: „Using a 3D Time-of-Flight Range Camera for Visual Tracking". In: Proceedings of IAV 2007, Toulouse.
- [8] Rohmoser, Bertram; Parlitz, Christopher: „Implementierung einer Bewegungsplanung für einen Roboterarm.“ In: Dillmann, Rüdiger (Tagungsleitung) u.a.; VDI/VDE-Gesellschaft Meß- und Automatisierungstechnik (GMA) u.a.: Robotik 2002: Leistungsstand, Anwendungen, Visionen, Trends. Tagung Ludwigsburg, 19. und 20. Juni 2002. Düsseldorf: VDI Verlag, 2002, S. 59-64 (VDI-Berichte 1679).
- [9] Sanchez, G.; Latombe, J.C.: "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. Int. Symposium on Robotics Research (ISRR'01)", Lorne, Victoria, Australia, November 2001.
- [10] Shreiner, Dave; Woo, Mason; Neider, Jackie: "OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2", Auflage: 3rd (2006), Addison-Wesley Longman, Amsterdam; Chapter 3: Viewing, Part "Overview: The Camera Analogy".
- [11] Song, Xin; Luo, Jun; Wang, Lu-Ping; Shen Zhen-Kang: "Ellipse Detection Approach Based on Convex Hull" in *Opto-Electronic Engineering*, 2007/10, vol. 34, pp. 40-44.
- [12] van Waveren, J.M.P: "From Quaternion to Matrix and Back", 2005, Id Software Inc.
- [13] Volz, René: "Entwurf und Implementierung einer schnellen Kollisionsdetektion für einen 7 DOF Roboter Manipulator", Diploma thesis at the University of Stuttgart.