Ein Ansatz zur Verbesserung der Posenbestimmung eines 3D-Kamerasystems basierend auf Referenzmessungen, K-Nächsten Nachbarn und einer Gewichtungsfunktion für Robotik-Anwendungen.

An approach to improve the pose estimation of a 3D camera system based on reference measurements, k-nearest neighbors and a weighting function for robotics applications.

Prof. Dr.-Ing. Rainer Müller, Dr.-Ing. Ali Kanso, Stefan Marx M.Sc., Xiaomei Xu M.Sc., Zentrum für Mechatronik und Automatisierungstechnik gGmbH, 66121 Saarbrücken, Deutschland, {rainer.mueller, a.kanso, s.marx, xiaomei.xu}@zema.de

Kurzfassung

In diesem Beitrag wird ein Ansatz zur Verbesserung der Positionsbestimmung einer 3D-Kamera vorgestellt, welche auf der Verwendung von Referenzmessungen und deren Verrechnung mittels der K-Nächsten Nachbarn-Methode (KNN) sowie einer Gewichtungsfunktion basiert. Damit konnte der Fehler bei der Positionsbestimmung der ausgewählten 3D-Kamera von 15-40 mm auf 1-3 mm reduziert werden. Grundlage für das Konzept war dabei die genaue Untersuchung der Ursachen für Messungenauigkeiten des Kamerasystems, wobei vor allem Abhängigkeiten mit der Entfernung des betrachteten Objekts sowie dem Neigungswinkel der optischen Achse der Kamera bezüglich der Objektoberfläche bzw. Oberflächennormalen festgestellt wurden. Untersucht und Validiert wurde der Ansatz anhand von Messungen mit LEGO-Bauteilen, welche in verschiedenen Positionen und Orientierungen im Sichtfeld der Kamera plaziert wurden. Die Referenzmessungen für die spätere Korrektur der Positionsdaten wurden mit einem Industrieroboter in Form von taktilen Messungen vorgenommen.

Abstract

This paper presents an approach to improve the position determination of a 3D camera, which is based on the use of reference measurements and their compensation by means of the k-nearest neighbour method and a weighting function. This reduced the error in the position determination of the selected 3D camera from 15-40 mm to 1-3 mm. The concept was based on a detailed investigation of the causes for measurement inaccuracies of the camera system, whereby above all dependencies with the distance of the observed object as well as the tilt angle of the optical axis of the camera with respect to the normal vector of the object surface were determined. The approach was investigated and validated using measurements with LEGO components placed in different positions and orientations in the camera's field of view. The reference measurements for the subsequent correction of the position data were made with an industry robot in the form of tactile measurements.

1 Einleitung

Die Erkennung von Objekten und die 3D-Posenbestimmung ist für die Bin Picking-Applikation unerlässlich [1]. Bei der Entwicklung flexibler und intelligenter Robotersysteme, kommen hierbei immer häufiger Kamerasysteme zum Einsatz, welche die Roboterumgebung erfassen sowie Objekte und Merkmale detektieren und lokalisieren [2]. Industriekameras können hierfür eine zehntelmillimetergenaue Positionsbestimmung bereitstellen [1], sind aber gleichzeitig teuer, zeitaufwändig in der Entwicklung für verschiedene industrielle Anwendungen, verfügen nicht über ausreichende öffentlich zugängliche Dokumentationen und erfordern deshalb zusätzliche kostenintensive Produktschulungen, um die teils komplexe Anwendungersoftware der Industriekameras effektiv verwenden zu können. Dieser Stand der Technik steht im Widerspruch zum Interesse kleiner und mittlerer Unternehmen an einer preisgünstigen, einfach zu entwickelnden und genauen 3D-Positionsbestimmung.

In diesem Beitrag wird eine 3D-Kamera vom Modell Azure Kinect DK verwendet, die in Universitätslabors [2]und kleinen und mittleren Unternehmen weit verbreitet ist. Für diese Kamera wurden verschiedene experimentelle Szenarien entwickelt um den Fehler bei der Positionsbestimmung für einen bestimmten Pixel anhand von umliegenden Referenzmessungen zu korrigieren. Für aussagekräftige Referenzmessungen wurden dabei verschiedene Einflussfaktoren vor allem auf die Tiefenmessung des verbauten Infrarotsensors berücksichtigt. Eine starke Korrelation konnte dabei zwischen der dem Neigungswinkel der Objektoberfläche bzgl. der Kamera und der Größe des Messfehlers bei der Tiefenmessung ermittelt werden.

2 Stand der Technik

In den folgenden Unterkapiteln werden kurz die Haupteinsatzgebiete der in diesem Beitrag verwendeten Azure Kinect DK sowie die Einflussfaktoren bei der 3D-Positionsbestimmung bei dieser Kameraart.

2.1 Haupteinsatzgebiete von Azure Kinect DK

Mit dem Durchbruch in der Tiefensensorik und der Technologie des maschinellen Lernens [3] sind tragbare, preisgünstige und einfach zu entwickelnde Alternativen für die 3D-Bewegungsanalyse kommerziell verfügbar geworden [4]. Azure Kinect DK (**D**evelop **k**it) umfasst einen 12-Megapixel-RGB-Sensor und einen 1-Megapixel-Tiefensensor und beinhaltet weitere Sensoren, die Anwendungen in der Gesichts- und Spracherkennung sowie im Hand- und Body-Tracking unterstützen [5, 6]. Ein Tiefenbild der Kamera kann beispielsweise verwendet werden, um die Hand- oder Körperregion zu segmentieren. [7] [8].

2.2 Einflüsse auf die 3D Positionsbestimmung

Die Hauptgründe für die Ungenauigkeiten bei der Positionsbestimmung mit 3D-Kameras sind [9]:

1. Der Tiefensensor und der RGB-Sensor haben unterschiedliche Koordinatensysteme.
2. Der Tiefensensor und der RGB-Sensor haben unterschiedliche Auflösungen, was die Transformation des Tiefenbildes in das RGB-Bild beeinflusst.
3. Der Tiefensensor und der RGB-Sensor weisen bei der Umwandlung ineinander tote Winkel auf.

Umgebungsgeräusche, sowie die IR-Beleuchtungsmasken der Kamera, als auch gesättigte oder schwache Infrarotsignale und Reflexionen beeinflussen die Ungenauigkeit des Tiefensensors [10]. Die wesentlichen Einflussfaktoren auf die Tiefenbilder sind außerdem die Temperatur, der Abstand von Objekten zur Kamera oder die Farbe der Szene [11].

3 Vorbereitungen für die 3D Posenbestimmung

Um, wie in Kapitel 4.3 beschrieben, eine Beziehung zwischen dem Neigungswinkel θ und dem Fehler $\Delta^C Z_p$ zu erhalten, wird die Kamera um die X-Achse, die Y-Achse oder auch um beide Achsen gleichzeitig bezüglich des Kamerakoordinatensystems in 5-Grad-Schritten von 0 Grad bis 45 Grad gedreht. θ bezeichnet hierbei den Winkel zwischen der Oberflächennormalen des betrachteten Objekts und der optischen Achse der Kamera, $\Delta^C Z_p$ der Fehler bei der Posenbestimmung entlang der Z-Achse (optische Achse) der Kamera. Bedingt durch den Aufbau der Roboterzelle und des Versuchsaufbau können die bei gleizeitiger Drehung um die X- und Y-Achse nur Drehwinkel bis 30° eingestellt werden. Die LEGO-Bauteile, die bei diesen Referenzmessungen als Target benutzt werden, werden in drei Reihen und fünf Spalten eben auf dem Tisch platziert. Für jeden Winkel werden nun ein RGB-Bild mit der Auflösung 3840x2160 und ein Tiefenbild mit der Auflösung 1280x720 extrahiert, sowie jeweils ein Bild eines auf der Tischebene plazierten Schachbrettmusters, welches auch zur Kamerakalibrierung verwendet wird. Der Fehler der Position jeder LEGO Bauteil wurde dann mit der Methode in Abschnitt 4.1 berechnet und als Referenzwert für die 3D-Positionsfehlerkorrektur verwendet. Das $\Delta^C Z_p$ jeder Position in Bezug auf den Neigungswinkel θ wurde gemäß der Methode in Abschnitt 4.2 extrahiert, und die Beziehung zwischen den beiden Parametern anschließend mit einer Trendlinie angenähert.

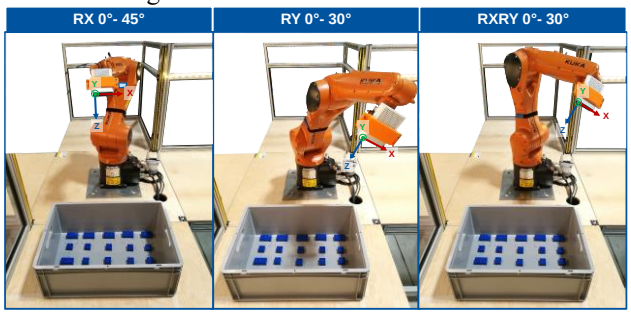

Abbildung 1 Position und Orientierung der Kamera über den LEGO-Bauteilen zur Suche nach der Korrelation zwischen Neigungswinkel und Tiefenfehler. Links) Rotation um TCP-X-Achse; Mitte) Rotation um TCP-Y-Achse; Rechts) Rotation um TCP-X- und TCP-Y-Achse

4 3D- Posen-Bestimmung

Abb. 2 zeigt einen Überblick zur Verbesserung der Positionsgenauigkeit mit K-Nächsten Nachbarn (KNN), und einer Gewichtungsfunktion. Wie bereits beschrieben, wird eine starke Korrelation zwischen dem Neigungswinkel θ und dem Tiefenfehler $\Delta^C Z_p$ vermutet, aber auch bzgl. der gewählten Rotationsachse des Kamerakoordinatensystems, um die die betrachtete Objektoberfläche rotiert ist.

4.1 Fehler des benachbarten LEGO Bauteils als Referenz für die Korrektur der 3D-Position des neuen LEGO Bauteils

Bevor ein Überblick über die 3D-Posen-Betimmungsmethode gegeben wird, sollte ein Experiment erwähnt werden, bei dem die Fehler der umgebenden LEGO-Bauteile als Referenz dienen, um die 3D-Position der neuen LEGO Bauteile vom Kamerasystem zu korrigieren.
Hierzu ist zunächst eine 2D-Kamerakalibirierung nötig, deren Zweck hauptsächlich darin besteht die Transformationsmatrix ${}^C\boldsymbol{T}_F$ vom Flanschkoordinatensystem F zum Kamerakoordinatensystem C zu bestimmen. Hierfür werden mit der

Abbildung 2 Verbesserung der Positionsgenauigkeit mit K Nächsten Nachbarn, Gewichtfunktion und die Trendline zwischen Neigungswinkel Rotationsachse und Tiefenfehler

am Roboterflansch montierten Kamera 21 Bilder von einem Schachbrettmuster der Größe 9x7 in der Auflösungen 3840x2160 aus verschiedenen Posen aufgenommen. Dabei ist zu beachten, dass die Kamera etwa 30 Sekunden benötigt, um sich an die Beleuchtung anzupassen und den Fokus einzustellen, wenn der Roboterarm zu einer neuen Pose bewegt wird. Die Qualität des Fotos wirkt sich stark auf die Genauigkeit der Kamerakalibrierung aus. Die OpenCV [12] (**Open** Source **C**omputer **V**ision Library) Kamerakalibrierungsoperation gibt eine 3x3 Kameramatrix und Verzerrungskoeffizienten für die intrinsischen Parameter, sowie den Translationsvektor ${}^{C}r_{CB}$ und den Rotationsvektor ${}^{C}t_{CB}$ in Rodrigues-Form vom Kamerakoordinatensystem zum Schachbrerttkoordinatensystem zurück (siehe Abb. 3). Translationsvektor und Rotationsvektor können anschließend zu der 4x4 Transformationsmatrix ${}^{C}\boldsymbol{T}_{CB}$ umgewandelt werden.

Die Pixelkoordinate $p = (u_p, v_p)$ können gemäß der folgenden Gleichung (1) in die Kamerakoordinaten $P = ({}^{c}X_p, {}^{c}Y_p, {}^{c}Z_p)$ transformiert werden [12]:

$$\begin{pmatrix} {}^{c}X_P/{}^{c}Z_p \\ {}^{c}Y_P/{}^{c}Z_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} \qquad (1)$$

Weiterhin werden die Transformationsmatrizen ${}^{F}T_B$ und ${}^{C}T_F$ benötigt. ${}^{F}T_B$ beschreibt die Transformation vom Roboterbasiskoordinatensystem des Roboters zum Roboterflansch und kann aus der aktuellen Position im KUKA smartPAD gewonnen werden. ${}^{C}\boldsymbol{T}_F$ ist die Transformationsmatrix vom Roboterflansch zur Kamera, deren Berechnung in [13] beschrieben wird.

Die Transformationsmatrix ${}^{C}\boldsymbol{T}_B$ vom Roboterbasiskoordinatensystem des Roboters zur Kamera ist die folgende, wobei ${}^{C}T_F$ die Transformationsmatrix von Flansch zu Kamerakoordinaten ist [13].

$${}^{C}\boldsymbol{T}_B = {}^{C}\boldsymbol{T}_F \cdot {}^{F}\boldsymbol{T}_B \qquad (2)$$

Der Roboterarm bewegt sich mit dem TCP (**T**ool **C**enter **P**oint) der am Flansch montierten Messspitze (Abb. 3) zur Mitte jeder LEGO-Bauteile, um die Sollwerte der Roboterkoordinaten $({}^{B}X_{P\_soll},\ {}^{B}Y_{P\_soll},\ {}^{B}Z_{P\_soll})$ bezüglich des Roboterbasiskoordinatensystems zu bestimmen. Die Vermessung des TCP der Spitze erfolgte dabei mit einem Absolutfehler von 0,254mm. Mit Gleichung (2) werden die Istwerte bezüglich des Kamerakoordinatensystems $({}^{C}X_{P\_soll},\ {}^{C}Y_{P\_soll},\ {}^{C}Z_{P\_soll})$ berechnet:

$$\begin{pmatrix} {}^{C}X_{P\_soll} \\ {}^{C}Y_{P\_soll} \\ {}^{C}Z_{P\_soll} \\ 1 \end{pmatrix} = {}^{C}\boldsymbol{T}_B \cdot \begin{pmatrix} {}^{B}X_{P\_soll} \\ {}^{B}Y_{P\_soll} \\ {}^{B}Z_{P\_soll} \\ 1 \end{pmatrix} \qquad (3)$$

Vergleicht man die Sollwerte mit den Istwerten, ermittelt über den RGB-Sensor und den Infrarotsensor, so ergibt sich der Fehler $(\Delta^C X_p,\ \Delta^C Y_P\ \ \Delta^C Z_P)$ zwischen ihnen:

$$\begin{pmatrix} {}^{C}X_{P\_soll} \\ {}^{C}Y_{P\_soll} \\ {}^{C}Z_{P\_soll} \end{pmatrix} - \begin{pmatrix} {}^{C}X_{P\_ist} \\ {}^{C}Y_{P\_ist} \\ {}^{C}Z_{P\_ist} \end{pmatrix} = \begin{pmatrix} \Delta^C X_P \\ \Delta^C Y_P \\ \Delta^C Z_P \end{pmatrix} \qquad (4)$$

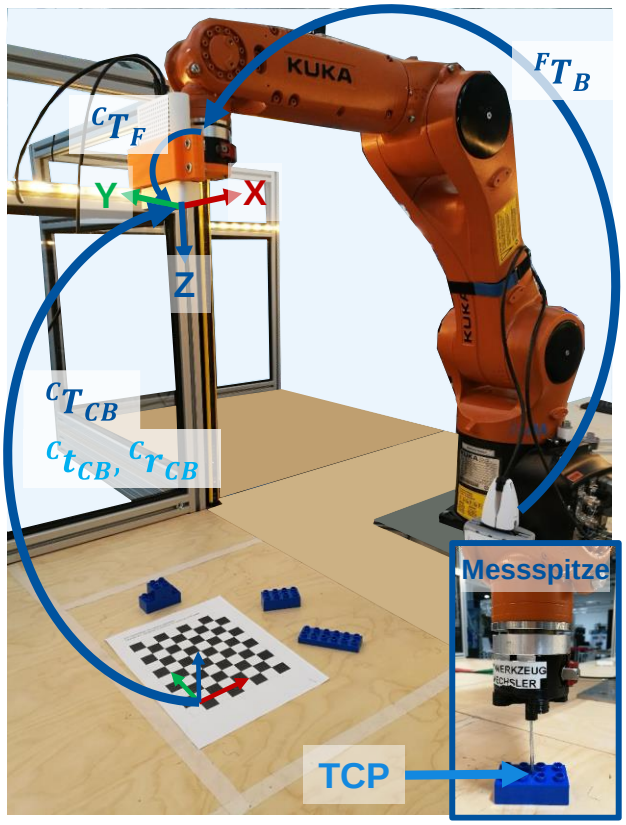

Abbildung 3 Koordinatensysteme und Koordinaten-transformationen im Versuchsaufbau

Der Fehler der LEGO-Bauteile ($\Delta^C X_P$, $\Delta^C Y_P$ $\Delta^C Z_P$) mit bekannter Sollposition wird nun als Referenz für die Korrektur neuer LEGO-Bauteile mit noch unbekannter Position im Raum bezüglich auf de Kamerakoordinatensystems verwendet.

Unter der Annahme, dass ein neues LEGO-Bauteil mit unterschiedlicher Höhe zufällig auf der Arbeitsfläche platziert wird, wird der Fehler in der Umgebung als Referenzwert für eine genaue 3D-Posen-Bestimmung verwendet. Die tatsächliche Position der neuen LEGO-Bauteile $({}^cX_{p\_new\_ist}, {}^cY_{p\_new\_ist}, {}^cZ_{p\_new\_ist})$ wird mit Hilfe von Gleichung (1) berechnet. Diese wird dann wird anhand der Fehler der umliegenden LEGO Bauteile aus den Referenzmessungen zur der Position $({}^cX_{p\_new\_cor}, {}^cY_{p\_new\_cor}, {}^cZ_{p\_new\_cor})$ korrigiert:

$$\begin{pmatrix} {}^cX_{P\_new\_cor} \\ {}^cY_{P\_new\_cor} \\ {}^cZ_{P\_new\_cor} \end{pmatrix} = \begin{pmatrix} {}^cX_{P\_new\_ist} \\ {}^cY_{P\_new\_ist} \\ {}^cZ_{P\_new\_ist} \end{pmatrix} + \begin{pmatrix} \Delta^C X_P \\ \Delta^C Y_P \\ \Delta^C Z_P \end{pmatrix} \quad (5)$$

Diese können wie folgt ins Roboterbasiskoordinatensystem konvertiert werden:

$$\begin{pmatrix} {}^BX_{P\_new\_cor} \\ {}^BY_{P\_new\_cor} \\ {}^BZ_{P\_new\_cor} \\ 1 \end{pmatrix} = {}^BT_F \cdot {}^FT_C \cdot \begin{pmatrix} {}^cX_{P\_new\_cor} \\ {}^cY_{P\_new\_cor} \\ {}^cZ_{P\_new\_cor} \\ 1 \end{pmatrix} \quad (6)$$

In Abschnitt 4.2 werden für diesen Fall passende Referenzwerte aus den umgebenden LEGO-Bauteilen mit Hilfe der KNN- und Gewichtsfunktionsmethode extrahiert.

4.2 K-Nächste Nachbarn und Gewichtungsfunktion

Die Suche nach den nächsten Nachbarn für jedes neue LEGO-Bauteil kann wie folgt beschrieben werden (siehe Abb. 2b). Für die nächstliegenden (nächsten Nachbarn) LEGO-Bauteile aus den Referenzmessungen wird der Abstand der Pixelkoordinaten $d_1, d_2, \dots, d_n$ zu einem neuen LEGO Bauteil berechnet. Die Gewichtung jedes dieser LEGO Bauteile ist dabei proportional zum Abstand und wird wie folgt berechnet:

$$w_1 = \frac{d_1}{\sum d_{sum}}, w_2 = \frac{d_2}{\sum d_{sum}}, \dots, w_n = \frac{d_n}{\sum d_{sum}} \quad (7)$$

$$\sum d_{sum} = d_1 + d_2 + \cdots + d_n \quad (8)$$

Die Idee dahinter ist, dass die näheren Nachbarn einen größeren Einfluss auf die Korrektur des neuen LEGO-Bauteils haben. Der Korrekturwert mit Hilfe der Gewichtsfunktion wird nun folgendermaßen berechnet (siehe Abb. 2 c):

$$\begin{pmatrix} \Delta^C X_{new} \\ \Delta^C Y_{new} \\ \Delta^C Z_{new} \end{pmatrix} = w_1 \cdot \begin{pmatrix} \Delta^C X_1 \\ \Delta^C Y_1 \\ \Delta^C Z_1 \end{pmatrix} + \cdots + w_n \cdot \begin{pmatrix} \Delta^C X_n \\ \Delta^C Y_n \\ \Delta^C Z_n \end{pmatrix} \quad (9)$$

Die berechneten Korrekturwerte ($\Delta^C X_{new}$, $\Delta^C Y_{new}$ $\Delta^C Z_{new}$) werden in die Gleichung (5) eingesetzt, um somit die Istwerte n $({}^cX_{p\_new\_ist}, {}^cY_{p\_new\_ist}, {}^cZ_{p\_new\_ist})$ anzupassen. Die Transformation dieser Koordinaten in das Roboterbasiskoordinatensystem ergibt sich wieder durch Einsetzen in Gleichung (6).

4.3 Korrelation zwischen dem Oberflächenneigungswinkel θ und dem Fehler $\Delta^C Z_P$

Bei industriellen Anwendungen im Bereich Bin Picking sind die zur greifenden Bauteile zufällig in beliebigen Orientierungen und Höhen in einer Kiste platziert. Wenn der Neigungswinkel der Oberflächennormalen des Bauteils bzgl. der optischen Achse des Kamerakoordinatensystems groß ist, ist die einfache K-Nächste Nachbarn-Methode und die Gewichtungsfunktion nicht ausreichend, um die mit Hilfe der Kamera bestimmten Position des jeweiligen Objektes zu korrigieren.

Dies wird an den folgenden Versuchen deutlich: Die Fehler an den Positionen 3b, 5b, 11b, 13b (siehe Abb. 2b) dienen als Referenz, um die Fehler an den Positionen 2a, 3a, 8a, 9a (siehe Abb. 2a) zu korrigieren. Die Fehler an diesen Positionen werden dabei hauptsächlich durch hohe Fehler $\Delta^C Z_P$ aus der Tiefenwertbestimmung beinflusst (siehe Tabelle 1). Die Fehler $\Delta^C Z_P$ der Referenzmessungen waren hingegen deutlich kleiner (siehe Tabelle 2), was dazu führt, dass die Referenzmessungen in dieser Art nicht ausreichen um den Tiefenwert zu korrigieren. Dies wird z.B. an Position 3b (Abb. 2(b)) deutlich, dessen Fehler $\Delta^C Z_P$ -11.54mm beträgt, wohingegen für den nächsten Nachbarn (Position 5a aus der Referenzmessung (Abb. 2(a))) nur -2.98mm gemessen wurden. Die Fehler in ^{C}X und ^{C}Y hingegen können gut durch die Referenzmessungen ausgeglichen werden, und erhalten zusätzlich eine Korrektur durch angepasste ${}^cZ_P$-Werte auf Grund des Zusammenhangs aus Gleichung (1). Die Oberflächen LEGO Bauteile an den Positionen 2a, 3a, 8a, 9a sind um einen großen Winkel θ gekippt. Es wird daher versucht, die Korrelation zwischen dem Kippwinkel θ und $\Delta^C Z_P$zu finden.

Tabelle 1 Positionsfehler der zufällig orientierten LEGO-Bauteile für ausgewählte Positionen ohne Korrektur in Abb. 2a)

Pos.	2a	3a	8a	9a
$\Delta^C X_P$	4,55mm	2,09mm	12,52mm	6,91mm
$\Delta^C Y_P$	5,67mm	4,02mm	1,319mm	1,52mm
$\Delta^C Z_P$	-4,77mm	-9,36mm	-11,54mm	-11,83mm

Tabelle 2 Positionsfehler der Referenzmessungen zu ausgewählten Positionen aus Abb. 2b)

Pos.	3b	5b	11b	13b
$\Delta^C X_P$	4,91mm	2,33mm	13,69mm	6,58mm
$\Delta^C Y_P$	4,38mm	3,43mm	5,13mm	4,15mm
$\Delta^C Z_P$	-1,03mm	-2,98mm	-3,06mm	-3,91mm

Der Winkel θ wird dabei durch die folgende Methode ermittelt: Zunächst wird ein Normalenvektor ${}^C\underline{n}$ für die Oberfläche des entsprechenden LEGO-Bauteils bestimmt. Hierzu werden die 3D-Positionen $P_1, \dots, P_n$ einer Vielzahl

von Punkten auf der Oberfläche des LEGO-Bauteils bestimmt. Als nächstes wird nun der Mittelwert aller verwendeten Punkte nach folgender Gleichung berechnet.

$$P_m = \sum_{i=1}^{n} \frac{P_i}{n} \tag{10}$$

Dieser dient dann dazu, mit Hilfe der einzelnen Punkte $P_1, \dots, P_n$ eine Matrix der Form $A = (P_1 - P_m, \dots, P_n - P_m)^T$ aufzustellen. Damit lässt sich dann das in [14] beschriebene Minimierungsproblem zum Bestimmen eines Best-fit-Vektors aufstellen:

$$r = \left\| A \cdot {}^{C}\underline{n} \right\|^2 \stackrel{!}{=} \text{MIN} \tag{11}$$

Die Lösung dieses Problems führt zu folgender Gleichung, wobei die Matrix V mit Hilfe der Sigularwertzerlegung berechnet werden kann [14]:

$${}^{C}\underline{n} = V \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{12}$$

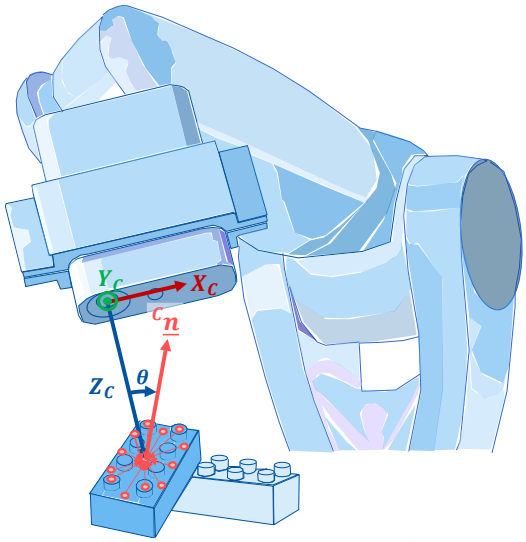

Abbildung 5 Bestimmung des Normalenvektors ${}^{C}\underline{n}$ und des Neigungswinkels θ

Der Winkel θ kann nun als der Winkel zwischen dem Normalenvektor ${}^{C}\underline{n}$ sowie der Z-Achse ${}^{C}\underline{Z_C}$ des Kamerakoordinatensystems mit Hilfe der folgenden Gleichung bestimmt werden.

$$\theta = arccos({}^{C}\underline{n}^T \cdot {}^{C}\underline{Z_C}) \tag{13}$$

Die Rotationsachse ${}^{C}\underline{V}$, um die mit dem Neigungswinkel θ gedreht wird, wird wie folgt berechnet:

$${}^{C}\underline{V} = {}^{C}\underline{n} \times {}^{C}\underline{Z_C} \tag{14}$$

Durch Normieren dieses Vektors erhält man den Einheitsnormalvektor ${}^{C}\underline{v} = {}^{C}\underline{V}/|{}^{C}\underline{V}|$

5 Implementierung

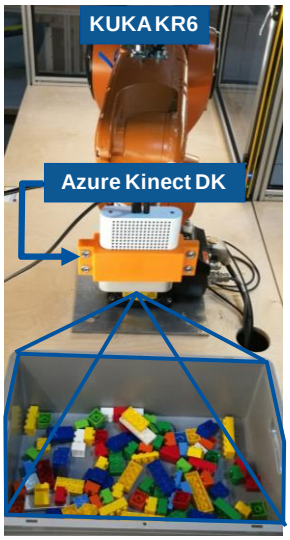

Abbildung 4 3D-Posen-Bestimmung mit RGB-Sensor und Tiefensensor

Für den Versuchsaufbau wurde die Kamera am Roboterflansch montiert (Abbildung 5). Die LEGO Bauteile werden in einem beliebigen Neigungswinkel in einer Kiste plaziert und die Kamera in beliebiger Höhe darüber ausgerichtet. Unterschiedlich ausgerichtete LEGO-Bauteile oder eine anders ausgerichtete Kamera haben dabei den gleichen Effekt. Beides führt zu unterschiedlichen Winkeln θ zwischen der Z-Achse des Kamerakoordinatensystems ${}^{C}\vec{Z}_C$ und dem Normalenvektor auf den LEGO-Bauteilen. Um den Winkel zu bestimmen, wurde das in Kapitel 4.3 beschriebene Konzept umgesetzt. Die Genauigkeit des Infrarotsensors hat dabei eine hohe Korrelation zur Objektfarbe, wie bereits in einer vorangehenden Veröffentlichung analysiert und beschrieben [9]. Daher werden für jede Farbe die Fehlerwerte $(\Delta^C X_p, \ \Delta^C Y_P \ \ \Delta^C Z_P)$ benötigt, um einen Referenzwert für die Korrektur der Position der neuen LEGO-Bauteile zu erstellen. Die detaillierten Korrekturformeln sind in den Abschnitten 4.1, 4.2 und 4.3 beschrieben.

6 Validierung & Evaluation

In diesem Kapitel werden nun zunächst die Ergebnisse bzw. Verbesserungen bei der Positionsbestimmung unter Verwendung der KNN-Methode sowie der Geweichtungsfunktion beschrieben. Anschließend wird aufgezeigt, wie sich der Neigungswinkel θ auf die Positionsbestimmung auswirkt. Zuletzt wird beschrieben, wie das Konzept in einer zukünftigen Arbeit die Posenbestimmung für eine Bin Picking-Applikation verbessern kann.

6.1 Ergebnisse bei Verwendung von K-Nächsten Nachbarn und der Gewichtungsfunktion

Im Folgenden werden die Ergebnisse zu drei unterschiedlichen Posen der Kamera über den betrachteten LEGO-Bauteilen vorgestellt. Diese drei Posen werden in diesem Abschnitt nachfolgend nur noch HOME, RX20 und RY20 genannt. Die HOME-Position stellt die Ausgangsposition des Roboters in dieser Versuchsreihe dar und hat die TCP-Koordinaten X: 58,83 mm, Y: 660,83mm, Z: 771,25mm, RX: -40,93°, RY: -0,03°, RZ: -179,99° bzgl. des Roboterbasiskoordinantensystems. RX20 und RY20 basieren auf HOME, wobei diese durch Rotationen mit 20 Grad um die X- und Y-Achse des Kamerakoordinatensystems verändert wurden. Da die Auflösungen der RGB- und Infrarotbilder unterschiedlich sind, muss sichergestellt werden, dass alle

LEGO Bauteile vom Infrarotsensor erfasst werden (siehe Abb. 6). Wenn sich LEGO-Bauteile nicht im Sichtfeld des Infrarotsensors befinden, ist der von der Kamera zurückgegebene Tiefenwert 0. Der tatsächliche Wert der LEGO Bauteil außerhalb des Infrarotbereichs kann nicht gemessen werden. In der Ansicht RX20 sind die LEGO Bauteile auf den Positionen 4 und 8 außerhalb des Sichtfeldes des Infrarotsensors. (siehe Abb. 6 RX20)

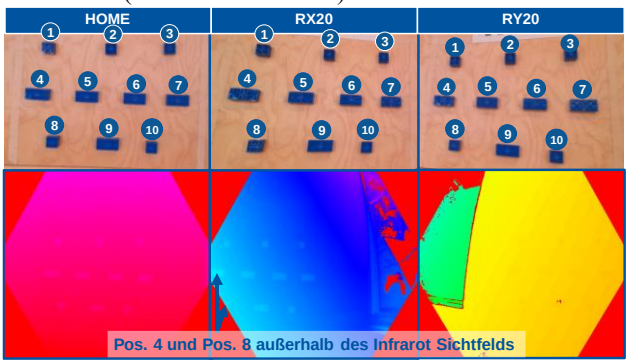

Abbildung 6 RGB-Bild und Tiefenbild von den LEGO-Bauteilen für die Positionen HOME, RX20 und RY20

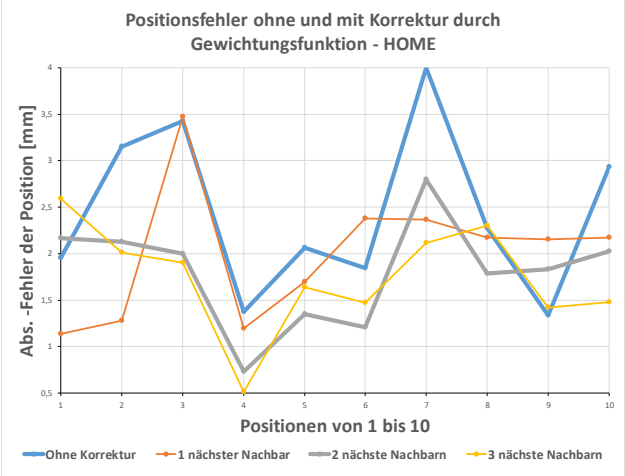

Abbildung 7 Absoluter Positionsfehler für die einezelnen Positionen in der HOME-Pose, ohne und mit Korrektur durch die Gewichtungsfunktion

Für die drei Posen HOME, RX20 und RY20 wurden dann der Positionsfehler für die unterschiedlichen LEGO-Bauteilen-Positionen 1-10 (Abb. 6) unter Berücksichtigung unterschiedlich vieler nächster Nachbarn ermittelt. Zur Korrektur der Werte wurden prinzipiell wieder Referenzmessungen nach dem Aufbau in Abb 2b durchgeführt. Neben Messungen aus der HOME-Pose wurden nun aber auch Referenzmessungen für die Posen RX20 und RY20 durchgeführt. Die Ergebnisse hierfür sind in den Diagrammen in Abb. 7-9 dargestellt. Über alle Positionen hinweg hat sich dabei die Korrektur mit zwei nächsten Nachbarn als die beste Lösung erwiesen. Im Falle der blauen LEGO-Bauteile konnte der Positionsfehler für Bilder auf aus der HOME-Pose auf etwa 3 bis 4 mm reduziert werden. Im günstigsten Fall konnten mit der KNN- und Gewichtsfunktions-Korrekturmethode auch Genauigkeiten von 1 bis 2 mm erreicht werden.

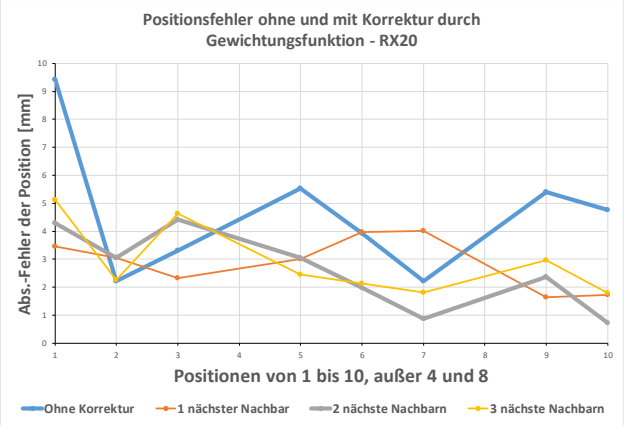

Abbildung 8 Absoluter Positionsfehler für die einezelnen Positionen in der RX20-Pose, ohne und mit Korrektur durch die Gewichtungsfunktion

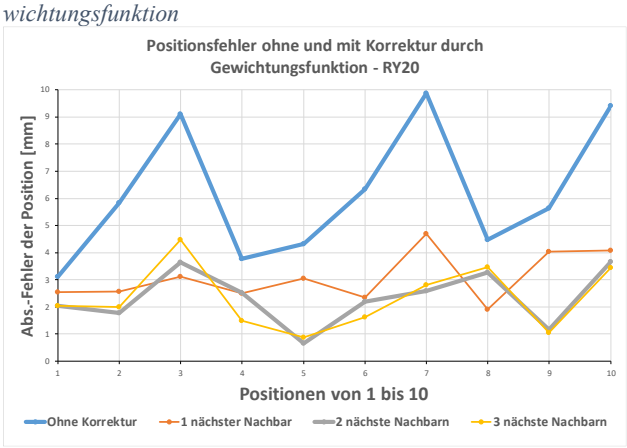

Abbildung 9 Absoluter Positionsfehler für die einezelnen Positionen in der RY20-Pose, ohne und mit Korrektur durch die Gewichtungsfunktion

6.2 Abhängigkeiten des Tiefenfehlers von der Orientierung des Bauteils

Um die Abhängigkeiten des Tiefenfehlers von der Orientierung des Bauteils zu untersuchen wurden Versuche, wie in Kapitel 3 beschrieben, mit unterschiedlichen Orientierungen bzw. Neigungswinkeln θ durchgeführt. Wird hierfür der Fehlerwert $\Delta^C Z_P$ über dem Neigungswinkel θ in einem Diagramm abgetragen (Abb. 10), so lässt sich ein deutlicher Trend des Werteverlaufs feststellen. Mit zunehmenden Winkel steigt auch der Fehlerwert nahzu linear an, weshalb versucht wird die Messdaten mit einer Trendlinie anzunähern. Weiterhin wurde auch untersucht, inwiefern die Rotationsachse ${}^C\underline{V}$ um welche die Kamera bzgl. der Objektoberfläche rotiert wird Einfluss auf den Fehler $\Delta^C Z_P$ hat. Das Ergebnis ist ebenfalls in Abb. 10 abgetragen. Wenn die Kamera sowohl um die X- als auch um die Y-Achse gedreht wird ergibt sich die Rotationsachse ${}^C\underline{V}$ allgemein in der Form ${}^C\underline{V} = (V_X, V_Y, 0)$. Auch entlang entlang einer solchen Rotationsachse lässt sich eine Korrelation mit dem Fehler $\Delta^C Z_P$, wie in Abb. 10 mit Hilfe der 3D-Trendlinie dargestellt, beobachten. Ziel ist die KNN- und Gewichtungsmethode zur Korrektur von $\Delta^C Z_P$ mit der Korrelation zum

Winkel θ zu kombinieren um die Positionsbestimmung auch für willkürlich Orientierte Bauteile zu verbessern.

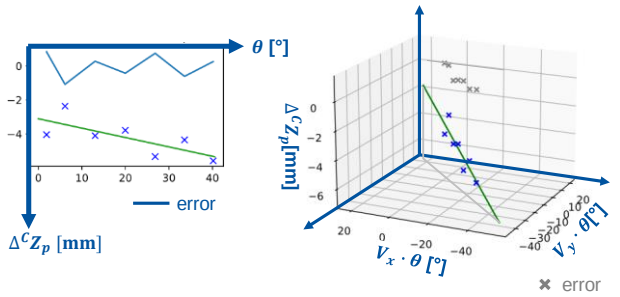

Abbildung 10 Die Korrelation zwischen Tiefenfehler und der Bauteilorientierung in Pos. 7 in Abb. 2 a)

Abbildung 11 Bilder der blauen LEGO Bauteile in einer Kiste für die Posen HOME, RX20 und RY20

Derzeit ist KNN- und Gewichtungsmethode ideal für die meisten unter der HOME Ansicht betrachteten Objekte, welche keinen oder nur einen geringen Neigungswinkel θ. aufweisen. Für größere Neigungswinkel um die X- und Y-Achse hingegen reicht die Verwendung dieser Methode nicht aus um gute Ergebnisse zu erzielen. (siehe Abb. 11 Positionen 3, 4, 7 und 8)

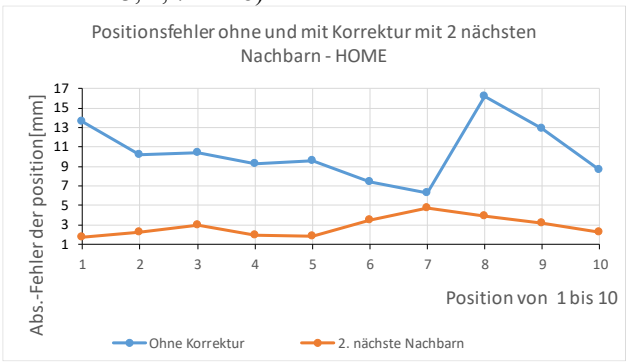

Abbildung 12 Absoluter Positionsfehler mit und ohne Korrektur für HOME-Pose (siehe Abb. 11 a)

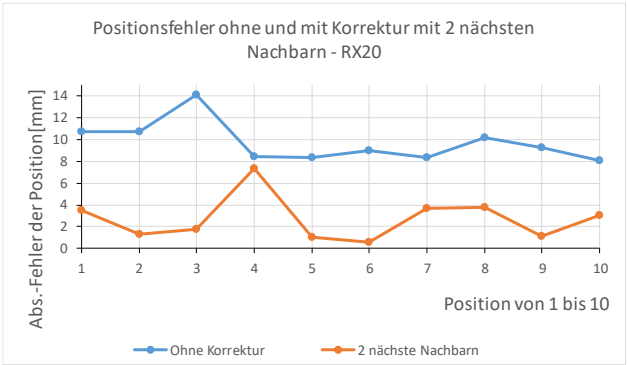

Abbildung 13 Absoluter Positionsfehler mit und ohne Korrektur für HOME-Pose (siehe Abb. 11 b)

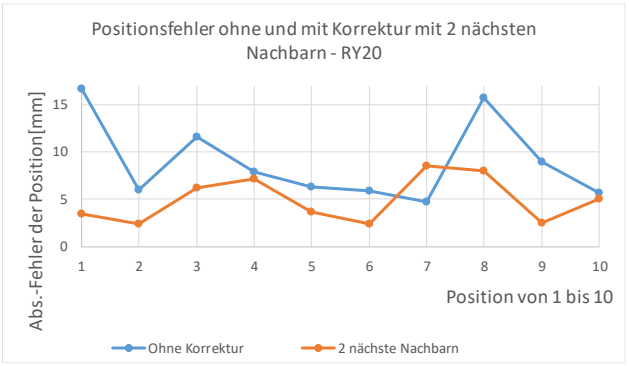

Abbildung 14 Absoluter Positionsfehler mit und ohne Korrektur für HOME-Pose (siehe Abb. 11 c)

6.3 Umsetzung für ein Bin Picking-Szenario

Die Validierung der Ergebnisse soll im nächsten Schritt auch an einem Bin Picking Szenario erfolgen. Hierbei geht es darum, willkürlich in einer Kiste angeordnete Objekte mit Hilfe einer Kamera zur erkennen und anschließend mit dem Roboter zu greifen. Die Posenbestimmung soll auch hierbei mit Hilfe der vorgestellten Azure Kinect DK erfolgen. Das im Folgenden vorgestellte Konzept dazu, beschäftigt sich mit dem Greifen rotationssysmetrischer Teile, und bedarf deshalb eines anderen Ansatzes zur Bestimmung eines Normalenvektors, wie zuvor mit der Best-fit-Methode und den Lego-Bauteilen beschrieben. Grundsätzlich wird dabei angenommen, dass die Bauteile senkrecht zu ihrer Rotationsachse mit einem Parallelgreifer gegriffen werden. Die Bauteilerkennung wird hierbei mit einer KI-basierten Methode umgesetzt, welche in einer weiteren Veröffentlichung des ZeMA beschrieben ist [15]. Die Kamera ist dabei nicht statisch montiert, sondern am Endeffektor befestigt. Dadurch müssen geringfüge Änderungen bei der Kamerakalibrierung vorgenommen werden, die grundsätzliche Methode bleibt aber identisch. Die Position eines Punktes P im Raum bzgl. des Roboterbasiskoordinatensystems B kann somit durch die nachfolgende Transformation beschrieben werden:

$$ {}^{B}\underline{r}_P = \begin{pmatrix} {}^{B}\underline{r}_C \\ 1 \end{pmatrix} = {}^{B}T_F \cdot {}^{F}T_C \cdot \begin{pmatrix} {}^{C}\underline{r}_P \\ 1 \end{pmatrix} \qquad (15) $$

Damit lässt sich bereits die Position des zu greifenden Objekts bestimmen. Der Wert ${}^{C}Z_P$ im Vektor ${}^{C}\underline{r}_P$ wird mit dem in Kapitel 4.2 beschriebenen Verfahren optimiert, was nachfolgend auch wieder Auswirkungen auf die X- und Y-Komponenten hat. Eine zum Greifen passende Orientierung eines rotationssymmetrischen Objekts lässt sich, wie bereits erwähnt, nicht mit der Best-fit-Methode bestimmen, weshalb ein weiteres Referenzkoordinatensystem T eingeführt wird. Dieses wird auf der Tischebene definiert, sodass dessen Z-Achse als die Flächennormale fungiert. Die Lage des Koordiantensystems ist dabei nicht relevant und sollte deshalb nach Möglichkeit mit drei Punkten aus Bereichen der Kamera konstruiert werden, für welche, mit dem in Kapitel 4.2 beschriebenen Verfahren, gute Genauigkeiten erzielt werden können. Über zwei Punkte P_1 und P_2 entlang der

Längsachse des Objekts kann nun eine Achse des Objektkoordinatensystems *P* berechnet werden.

$$ {}^{C}\underline{x}_P = \frac{{}^{C}\underline{r}_{P2} - {}^{C}\underline{r}_{P1}}{\left\| {}^{C}\underline{r}_{P2} - {}^{C}\underline{r}_{P1} \right\|} \tag{16} $$

Die Y-Achse soll nun senkrecht zu dieser X-Achse, sowie parallel zur X-Y-Ebene des Bezugskoordinatensystems *T* bestimmt werden. Dies kann mittels des Kreuzprodukts aus der X-Achse sowie der Flächennormalen bzw. der Z-Achse von *T* berechnet werden.

$$ {}^{C}\underline{y}_P = {}^{C}\underline{x}_P \times {}^{C}\underline{z}_T \tag{17} $$

Die fehlende Komponente der Objektorientierung berechnet sich dann zu

$$ {}^{C}\underline{z}_P = {}^{C}\underline{x}_P \times {}^{C}\underline{y}_P \tag{18} $$

Die so bestimmte Objektorientierung sorgt dafür, dass die Objektnormale ($ {}^{C}\underline{z}_P $) den kleinst möglichen Winkel zur Flächennormale der Tischebene hat. Dies bildet eine gute Ausgangslage für das Greifen der Objekte. Eventuell nötige Anpassungen in der Greiforientierung können nun durch einfache Rotationen um $ {}^{C}\underline{x}_P $ beschrieben werden.

Die Genauigkeit der Positionsbestimmung bei der Bestimmung der Objektorientierung spielt bei dem beschriebenen Ansatz eine große Rolle, da mindestens fünf einzelne Positionen (zwei auf dem Objekt und drei für das Referenzkoordinatensystem *T*) in die Berechnung mit eingehen. Mit bestimmten Greiferdesigns lassen kleinere Ungenauigkeiten beim Greifen nochmals ausgleichen, beim Greifen aus der Kiste ist der Arbeitsraum für Roboter und Greifer oftmals eingeschränkt weshalb größere Ungenauigkeiten bei der Posenbestimmung der Objekte zu Problemen beim Greifen führen können

7 Zusammenfassung

In diesem Paper wurde experimentelle Versuchreihe für eine 3D-Kamera vom Typ Azure Kinect DK vorgestellt, welche das Ziel hatte die eine Methode zur Verbesserung der Positionsbestimmung mit der Kamera herzuleiten. Dabei wurde festgestellt, die Positionsbestimmung für einen beliebigen Pixel/Objekt mit Hilfe von Referenzmessungen aus der nahen Umgebung (k nächste Nachbarn) dieses Pixels korrigiert werden können. Verrechnet werden diese Referenzmessungen dabei zusätzlich noch mit einer Gewichtungsfunktion, welche den direkten Abstand der Referenzen zum betrachteten Pixel/Objekt berücksichtigt. Gewichtsfunktion und die K-Nächste-Nachbar-Methode werden für die 3D Posenbestimmung unter verschiedenen Neigungswinkeln und verschiedenen Höhen ausprobiert. Hiefür wurde eine hohe Korrelation zwischen dem Neigungswinkel θ, der Rotationsachse $ {}^{C}\underline{V} $ und dem Fehler $ \Delta {}^{C}Z_P $, entlang der Z-Achse der Kamera gefunden. In Zukunft wird eine weitere Methode mit der Korrelation zwischen dem Neigungswinkel, der Rotationsachse und dem $ \Delta {}^{C}Z_P $ mit der Gewichtungsfunktion und K nächsten Nachbarn kombiniert, um die 3D Posenbestimmung ideal unter verschiedenen Neigungswinkeln und verschiedenen Höhen zu realisieren.

Es werden noch weitere Ansätze zur Identifikation der Orientierung des Bauteils untersucht. Zu diesem Zweck, wird die Fähigkeit LEGO-Bauteiloberflächen genau zu erkennen und Konturen klar zu erfassen in Zukunft weiterentwickelt. Auch stehen weitere Experimente zur Untersuchung des Einflusses der Objektefarbe bei 3D-Positionsbestimmung mit Hilfe der Infrarotsensortechnik aus.

8 Literatur

[1] Cognex, 3DA5000 Series Area Scan 3D Camera.

[2] M. Tölgyessy, M. Dekan, and Ľ. Chovanec, “Skeleton Tracking Accuracy and Precision Evaluation of Kinect V1, Kinect V2, and the Azure Kinect,” Applied Sciences, vol. 11, no. 12, p. 5756, 2021

[3] Steffi L., Colyer, Marray Evans, Darren P. Cosker, Aki I.T.Salo, “A Review of the Evolution of Vision-Based Motion Analysis and the Intgration of Advanced Computer Vision Methods Towards Developing a Markerless System,”

[4] L.-F. Yeung, Z. Yang, K. C.-C. Cheng, D. Du, and R. K.-Y. Tong, “Effects of camera viewing angles on tracking kinematic gait patterns using Azure Kinect, Kinect v2 and Orbbec Astra Pro v2,” *Gait & posture*, vol. 87, pp. 19–26, 2021

[5] A. Mamikonyan, M. Gevorgyan, M. Beyeler, *OpenCV 4 with Python Blueprints: Build Creative Computer Vision Projects with the Latest Version of OpenCV 4 and Python 3*, 2020.

[6] Microsoft, *Azure Kinect DK.*

[7] C. Keskin, F. Kırac¸, Y. E. Kara, and L. Akarun, “Real Time Hand Pose Estimation Using Depth Sensors,” *Springer*, 2013, pp. 119–137.

[8] V. Athitsos and S. Sclaroff, “Estimating 3d hand pose from a cluttered image,” *IEEE Computer Society Conference on Computer Vision*, 2003, p. 432.

[9] R.Müller, A.Kanso, H.Sheaib, X.Xu, “Development of a Real-time Observation System to Track Body/Hand Gestures for Industrial Robot Using Azure Kinect Camera,” Nov. 2021, pp. 58–65, 2021.

[10] Microsoft, *Tiefenkamera in Azure Kinect DK.*

[11] O. Wasenmueller D. Stricker, “Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision,” *Asian Conference Computer Vision*

[12] A. Kaehler and G. R. Bradski, *Learning OpenCV 3: Computer vision in C++ with the OpenCV library*. Beijing, Boston, Mass., Farnham, Sebastopol, Tokyo: O'Reilly, 2017.

[13] Müller, R., Vette-Steinkamp, M. u. Kanso, A., “Position and orientation calibration of a 2D laser line sensor using closedform least-squares solution,” *IFAC-PapersOnLine 52*, pp. 689–694.

[14] Best-fit method for the calibration of 3D objects using a laser line sensor mounted on the flange of an articulated robot, “A. Kanso, T. Masiak, M. Vette R.Müller,” *MHI-Fachkolloquium 2019*

[15] R. Müller, A. Kanso, S. Marx, and M. J. Islam, “A Concept for Object Detection and Localization based on AI and Computer Vision as a Basis for Bin Picking Applications,” in *RACIR 2021*, Eds., Birkenfeld: Shaker Verlag, 2021, pp. 28–35.

DOI: 10.17185/duepublico/75430
URN: urn:nbn:de:hbz:465-20220222-152058-2