

Applikationen mit Universal Robots und *VeriSens*® bei verschiedenen Höhen als auch relativen Objektpositionen

AN202005/v0.1/2020-12-17

Beschreibung

Lösen von Applikationen mit Universal Robots (UR) und *VeriSens*® bei verschiedenen Höhen als auch relativen Objektpositionen

Produkte

VeriSens® XF900 and XC900 series

Inhalt

1	Technischer Hintergrund.....	2
2	Variante 1 – einfach mehrere Jobs anlegen	2
2.1	Applikationen	2
2.2	Vor- und Nachteile der Variante.....	3
2.3	Installation und Kalibrierung	3
2.4	Bildverarbeitungsjob und Roboterprogramm	3
3	Variante 2 – vom Roboterprogramm beeinflusstes Anfahren	4
3.1	Applikationen	4
3.2	Vor- und Nachteile der Variante.....	4
3.3	Installation und Kalibrierung	4
3.4	Bildverarbeitungsjob und Roboterprogramm	5
3.5	Programmbeispiel 1 – Siegertreppchen	6
3.5.1	Beschreibung der Applikation	6
3.5.2	Anlegen der Variablen	6
3.5.3	Erstellen des Programms	7
3.6	Programmbeispiel 2 – vier Schrauben abhängig von variabler Position.....	10
3.6.1	Beschreibung der Applikation	10
3.6.2	Anlegen der Variablen	11
3.6.3	Erstellen des Programms	12
4	Zusammenfassung/Spezialfälle.....	17
5	Downloads	17
6	Support	17
7	Rechtliche Hinweise	17

1 Technischer Hintergrund

Bildverarbeitung mit *VeriSens*® beruht auf 2D-Bildern. Robotik bewegt sich im 3D-Raum. Der Einfluss der dritten Dimension auf die 2D-Skalierung wird beim Kalibrierprozess mittels *SmartGrid* bei der *VeriSens*® / UR Integration bereits eingelernt. Dies erleichtert bei der anschliessenden Einrichtung den Umgang mit den verschiedenen «Z» wie Objekthöhe, Greifhöhe, Höhe der Arbeitsfläche erheblich, da alle Werte so mit einer «Bezugsebene» arbeiten können.

Bei einem Objekt wird dessen Höhe gemessen und im Bildverarbeitungs-Job hinterlegt. Damit kann *VeriSens*® an den Roboter nicht nur die per 2D-Bildverarbeitung ermittelten *x*, *y* und *rotZ* übermitteln, sondern auch den Wert *z* der «gesehenen» Objektoberfläche.

Hinweis

Vereinfacht sind *rotX* und *rotY* Null, damit sind alle 3D-Koordinaten bedient.

Nun taucht die berechtigte Frage auf, ob man mit *VeriSens*® und UR auch verschieden hohe Objekte in einer Applikation bedienen kann.

Stellen Sie sich ein Siegertreppchen vor (Abbildung 1).

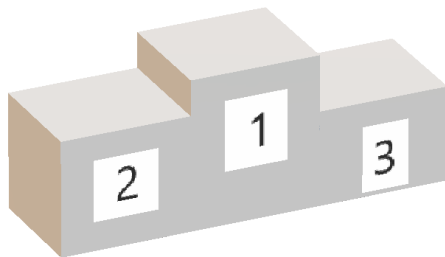


Abbildung 1: Siegertreppchen

Sie möchten per Bildverarbeitung von oben messen, ob die Stellflächen die richtigen Dimensionen haben.

Problem: Eine Bildaufnahme von oben aus einer Kameraposition führt dazu, dass die abgebildeten Stellflächen aufgrund der verschiedenen Distanzen zur Bildaufnahme unterschiedlich gross erscheinen.

Wie kann man dennoch solche Applikationen lösen?

2 Variante 1 – einfach mehrere Jobs anlegen

2.1 Applikationen

Ein Objekt mit verschiedenen Höhen soll mit wenig Aufwand geprüft bzw. vermessen werden. Dabei liegen alle Höhen im Schärfentiefebereich des *VeriSens*® in seiner kalibrierten Aufnahmeposition.

Im Beispiel unseres Siegertreppchens muss es also möglich sein, dass sich alle 3 Stufen im Schärfentiefebereich befinden. Die Schärfentiefe kann im Falle der *VeriSens*® XC-Serie durch die Auswahl eines entsprechenden Objektivs beeinflusst werden.

VeriSens® wird entsprechend der Technischen Dokumentation dynamisch montiert – also mitgeführt. Alternativ kann auch statisch, also über dem Roboter montiert werden.

2.2 Vor- und Nachteile der Variante

Vorteile:

- Sehr einfach zu verstehen und umzusetzen
- Mit beiden Montagevarianten realisierbar

Nachteile:

- Objekthöhen müssen in einem gemeinsamen Schärfentiefebereich liegen
- Konkrete Objekthöhen müssen bei Programmerstellung bekannt sein

2.3 Installation und Kalibrierung

Mittels *SmartGrid* wird die Applikation kalibriert. Der Fokus darf nun nicht mehr verstellt werden. Die Bildaufnahme-position wird mittels *VeriSens® URCap* im Schritt 2 gespeichert.

Hinweis

Achten Sie bitte darauf, dass bei der Eingabe der Distanz *SmartGrid*-Bezugsebene sowohl in der *VeriSens® Application Suite* als auch im *VeriSens® URCap* der zur *SmartGrid*-Materialdicke zusätzliche Aufschlag *SmartGrid* Unterseite – Bezugsebene berücksichtigt wird, wenn das *SmartGrid* nicht direkt auf der Bezugsebene liegt.

Nach der Durchführung des automatischen Koordinatenabgleichs im Installations-Schritt 3 des *VeriSens® URCap* kann die Applikation erstellt werden.

2.4 Bildverarbeitungsjob und Roboterprogramm

Für jede zu prüfende Höhe – in unserem Fall also die drei Höhen der Stellflächen – muss je ein *VeriSens®* Job angelegt bzw. angepasst werden («Stellfläche_1», «Stellfläche_2» etc.).

Dazu muss die jeweilige Objekthöhe (Höhe der Stufe zur Bezugsebene) in jedem Job unter Koordinaten / Z-Korrektur hinterlegt werden.

Da *VeriSens®* durch das Kalibrieren das Gesamtsystem kennt, ist er in der Lage, die aufgrund der Höhe unterschiedlich gross erscheinenden Objekte pro Job entsprechend passend in x und y zu skalieren und so die richtigen Masse zu überprüfen.

Wie könnte man die verschiedenen Höhen zusätzlich mit dem Roboter anfahren, z.B. um das Siegetreppchen per Roboter aus den einzelnen Stufen (Abbildung 2) zusammenzusetzen?

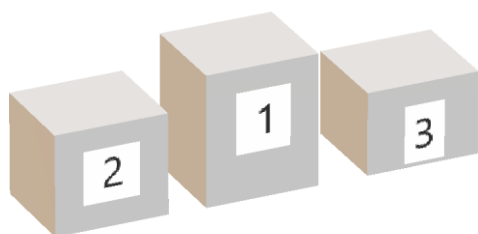


Abbildung 2: Einzelne Stufen eines Siegetreppchens

Ganz einfach, das Anfahren erfolgt aufgrund der Koordinaten der im Roboterprogramm jeweils vorgeschalteten «*VeriSens®* Job Ausführung» (EN: «*VeriSens®* Job Execution») unter Nutzung des für jede Stufe individuellen *VeriSens®* Jobs. Diese übergibt dem folgenden «*VeriSens®* Wegpunkt & Bewegen» (EN: «*VeriSens®* Waypoint & Move») die Koordinaten, einschliesslich z der Objekthöhe. Bei diesem Knoten kann

zusätzlich jeweils ein entsprechender Z-Offset frei gewählt werden und erlaubt so, dass der Greifer z.B. eine Position unterhalb der Oberfläche der Stufe anfährt.

Wenn die einzelnen Stufen zufällig bereitgestellt werden, könnte z. B. in Abhängigkeit des Ergebnisses eines Distanzsensors (Messung der Stufenhöhe) die Auswahl des dazugehörigen Jobs erfolgen.

3 Variante 2 – vom Roboterprogramm beeinflusstes Anfahren

3.1 Applikationen

- Ein durch einen Roboterarm geführter Schrauber soll vier Schrauben in den Ecken eines Objektes abhängig von einer durch VeriSens® gefunden Position anfahren.
- Ein mit dem UR mitgeführter Abstandssensor liefert die Distanz zum Objekt, mit der höhenabhängig und somit flexibel geprüft werden kann.
- Aus applikativen Gründen soll das Roboterprogramm selbst eine zusätzliche Abweichung (Offset) zu den von VeriSens® übermittelten bildbasierten Koordinaten in den Wegpunkt einfließen lassen.
- Bei unserem Beispiel des Siegertreppchens wird mit Variante 2 jede Stufe des Siegertreppchens individuell per Roboter angefahren und so mit immer gleichem Abstand zu VeriSens® vermessen.

Es ist ausschliesslich die dynamische Montage des VeriSens®, also das Mitführen mit dem Roboter möglich.

3.2 Vor- und Nachteile der Variante

Vorteile:

- Verschiedene Objekthöhen müssen nicht in einem gemeinsamen Schärfentiefebereich liegen
- Es ist ein Distanzsensor zur automatischen Ermittlung der Objekthöhe nutzbar
- Von Höhen z oder x , y , $rotZ$ abhängige Positionen lassen sich anfahren

Nachteile:

- Komplexer als Variante 1
- auf dynamische Montage mit deren Vor- und Nachteilen (siehe Doku) limitiert

3.3 Installation und Kalibrierung

In unserem Beispiel des Siegertreppchens wird die Höhe genau einer Stellfläche zur Bezugsebene gemessen und im Roboterprogramm fest vorgegeben. Das hat den Vorteil, dass nur ein Bildverarbeitungsjob nötig ist und man zudem unabhängig vom Schärfentiefebereich von VeriSens® arbeitet. Der für unterschiedlich hohe Stufen notwendige Aufschlag oder Abschlag der Koordinate z wird vom Roboterprogramm selbst vorgegeben.

Anhand der oberen Stufe unseres Siegertreppchens wird das System im VeriSens® *URCap* kalibriert, wozu das *SmartGrid* aufgelegt wird. Die Roboterposition bei dieser Bildaufnahme wird wie üblich unter Schritt 2 gespeichert. Dann erfolgt der Koordinatenabgleich in Schritt 3, der Focus VeriSens® wird nun nicht mehr verstellt.

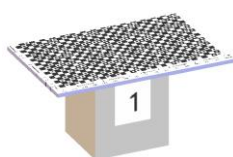


Abbildung 3: *SmartGrid* auf der Oberseite der höchsten Stufe

Hinweis

Achten Sie bitte darauf, dass bei der Eingabe der Distanz *SmartGrid*-Bezugsebene sowohl in der *VeriSens® Application Suite* als auch im *VeriSens® URCap* der zur *SmartGrid*-Materialdicke zusätzliche Aufschlag *SmartGrid* Unterseite – Bezugsebene berücksichtigt wird, da das *SmartGrid* nicht direkt auf der Bezugsebene liegt. Die Distanz *SmartGrid*-Bezugsebene ist somit die Summe aus *SmartGrid*-Materialdicke und Höhe der Stufe.

Für die Kalibrierung haben wir die obere Stufe gewählt, um nicht in Konflikte beim späteren Verfahren mit der Maximalhöhe zu kommen.

3.4 Bildverarbeitungsjob und Roboterprogramm

In unserem Beispiel des Greifens einzelner Stufen des Siegertreppchens wird ein Offset z für jede einzelne Stufe 1, 2 und 3 benötigt.

Der applikative Trick besteht nun darin, dass das Roboterprogramm selbst oder sogar ein Distanzsensor diesen zusätzlichen Offset z auf festgelegte Wegpunktkoordinaten oder bildbasierte Koordinaten legt, indem beide Koordinaten im Roboterprogramm verrechnet werden.

Konkret kommt der Offset z im Roboterprogramm zweimal zur Anwendung:

- 1. Höhenanpassung: Wegpunkt vor «*VeriSens®* Job Ausführung» (EN: «*VeriSens®* Job Execution»)
- 2. Höhenanpassung: Wegpunkt nach «*VeriSens®* Wegpunkt & Bewegen» (EN: «*VeriSens®* Waypoint & Move»)

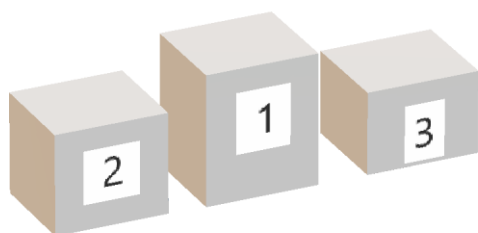


Abbildung 4: Einzelne Stufen des Siegertreppchens

1. Höhenanpassung

Für jede Stellfläche findet die selbe «*VeriSens®* Job Ausführung» (EN: «*VeriSens®* Job Execution») statt. Allerdings wird vor der Job Ausführung durch das programmabhängige Verfahren des Roboters ein fester Arbeitsabstand Oberseite Stufe – *VeriSens®* für jede der drei Stufen realisiert, so dass die dargestellte Stellfläche im Bild immer gleich gross ist und so richtig vermessen werden kann.

2. Höhenanpassung

Die von *VeriSens®* übermittelte bzw. im Bildverarbeitungsjob hinterlegte Koordinate z (unter Koordinaten / Z-Korrektur) ist immer gleich. Angeben wird die Höhe der Stufe 1 (Oberseite) zur Bezugsebene, da wir dafür einen Job eingerichtet haben.

Deshalb ist eine Script-basierte Berechnung im Roboter nötig, da erst im Roboter beide Sensoren bzw. Werte zusammenlaufen bzw. wird erst dort der zusätzliche, feste Offsetwert dem bildbasierten Wegpunkt aufgeschlagen.

Dazu muss das Script die Koordinaten des angefahrenen bildbasierten Wegpunktes zur weiteren Verrechnung erst übernehmen, was durch Anfahren einer Position mit festem Offset aus dem Knoten «*VeriSens®* Wegpunkt & Bewegen» (EN: «*VeriSens®* Waypoint & Move») erfolgt.

Der anzufahrende Wegpunkt für die Greifaufgabe des UR errechnet sich dann aus diesen so gespeicherten Koordinaten und dem erwähnten zusätzlichen Wert für die Höhenanpassung der jeweiligen Stufe.

Für Greifaufgaben – hier ist typischerweise eine Position unterhalb der Objektoberfläche erforderlich – wird weiterhin ein objektabhängiger Offset aus dem Roboterprogramm genutzt.

3.5 Programmbeispiel 1 – Siegertreppchen

3.5.1 Beschreibung der Applikation

Wir widmen uns hier dem Siegertreppchen (Abbildung 1) und zeigen, wie mit unterschiedlichen Objekthöhen z umgegangen wird, indem die im Bildverarbeitungs-Job vorgegebene Distanz zwischen *VeriSens*® und Objektoberfläche durch das Roboterprogramm selbst konstant gehalten wird.

3.5.2 Anlegen der Variablen

Die folgenden Variablen werden im Programm verwendet und müssen daher vorher angelegt werden (Abbildung 5). Die zunächst festgelegten Werte werden im laufenden Programm angepasst.

```
offs_exec_z
```

Vertikale Verschiebung der Bildaufnahme position gegenüber der Höhe beim Koordinatenabgleich (Installation).

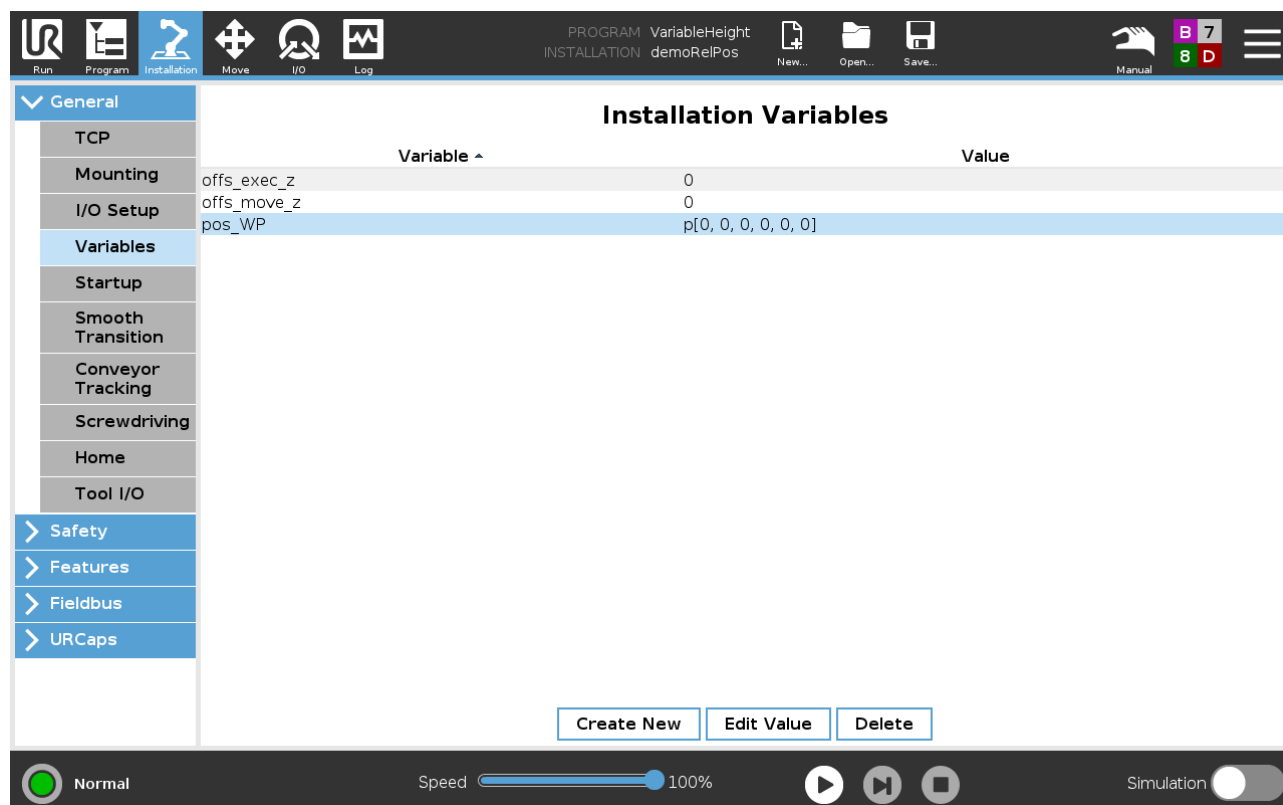
```
offs_move_z
```

Vertikale Verschiebung, die in *VeriSens*® Waypoint manuell eingestellt wurde.

```
pos_WP
```

Berechnete Position eines Wegpunkts, der im Programm angefahren werden soll.

Syntax: p[x, y, z, rotX, rotY, rotZ]



Variable	Value
offs_exec_z	0
offs_move_z	0
pos_WP	p[0, 0, 0, 0, 0, 0]

Abbildung 5: UR Controls, Installation Mode for Variables

3.5.3 Erstellen des Programms

Mit der Roboterprogrammierung kann nun begonnen werden (Abbildung 6).

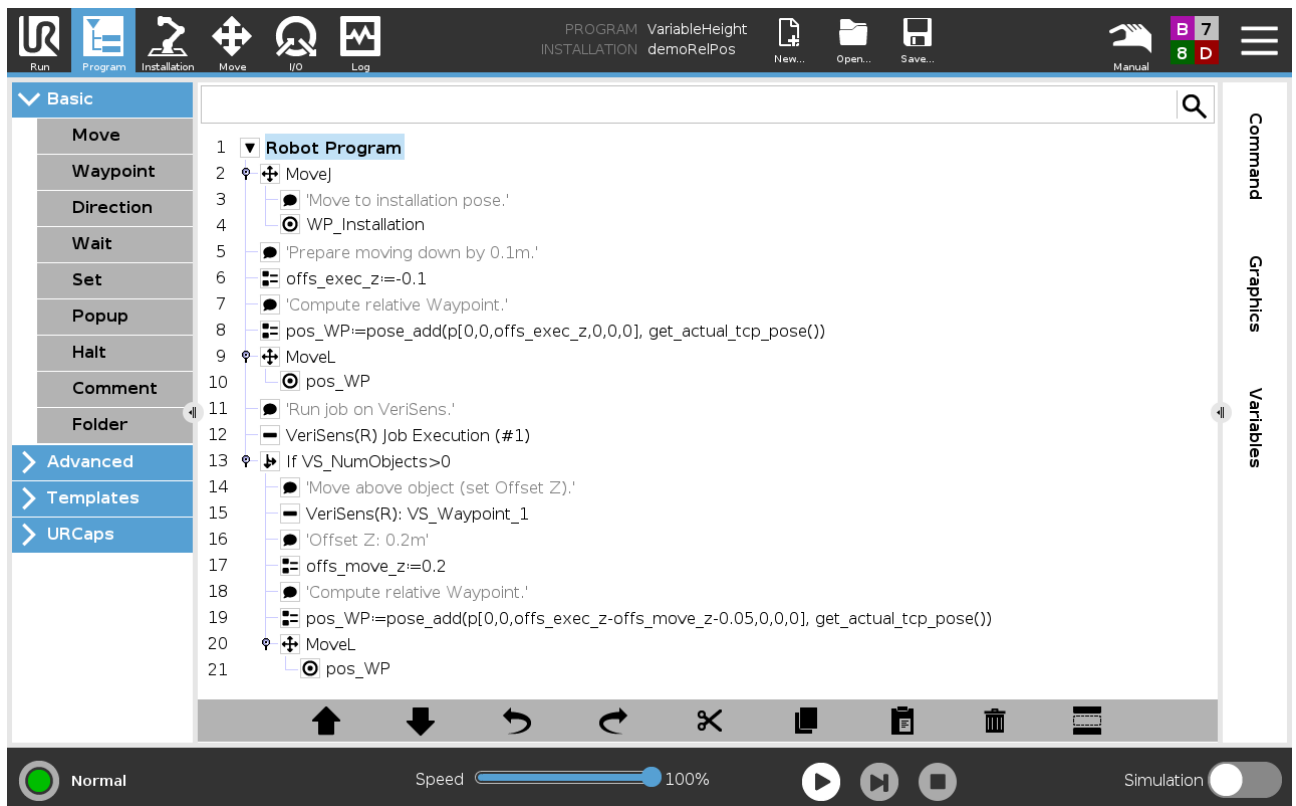


Abbildung 6: UR Controls, Programming

Programmbeschreibung

Zeile	Erläuterung
4	<p>→ Basic → Waypoint</p> <p>Fahre Wegpunkt an, der entweder der Pose des Vision Sensors während der Installation entspricht oder einer Pose mit derselben Höhe z wie bei der Installation.</p>
6	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>offs_exec_z</code> auf einen z-Offset, um den die Job Execution Pose verändert werden soll.</p> <p>Beispiel: Die aktuelle Inspektionsebene liegt 100 mm tiefer als die Pose bei der Kalibrierung (neg. Z-Richtung) => -0,1 Meter</p> <p>Alternativ könnte hier auch das Ergebnis eines externen Distanz-Sensors zur Anwendung kommen.</p>
8	<p>→ Advanced → Assignment</p> <p>1. Höhenanpassung: setze Variable <code>pos_WP</code> auf einen berechneten Wert. Dieser wird mit <code>pose_add()</code> als Summe der folgenden zwei Größen gebildet:</p> <ul style="list-style-type: none"> <code>p[0, 0, offs_exec_z, 0, 0, 0]</code> ... eine erzeugte relative Verschiebung in Z-Richtung <code>get_actual_tcp_pose()</code> ... die aktuelle Position des Roboters

Anmerkung:

- Eine Pose kann zur Verwendung in einer Formel mit folgendem Syntax erstellt werden: $p[x, y, z, \text{rotX}, \text{rotY}, \text{rotZ}]$. Dabei geben x, y, z die Koordinaten (z. B. metrisch in „Meter“) an und $\text{rotX}, \text{rotY}, \text{rotZ}$ die Rotationswerte des TCP (in Radianten).
- In diesem Beispiel bleiben bei berechneten Posen die letzten drei Komponenten (Rotationswerte) unangetastet.

Hinweis

Generell empfehlen wir, applikationsabhängig ausschliesslich rotZ zu beeinflussen, um ungewollte Seiteneffekte zu vermeiden, es wird auch nur dieser von *VeriSens*[®] geliefert.

- pos_WP dient als vorübergehend genutzte Variable zur Speicherung einer Position

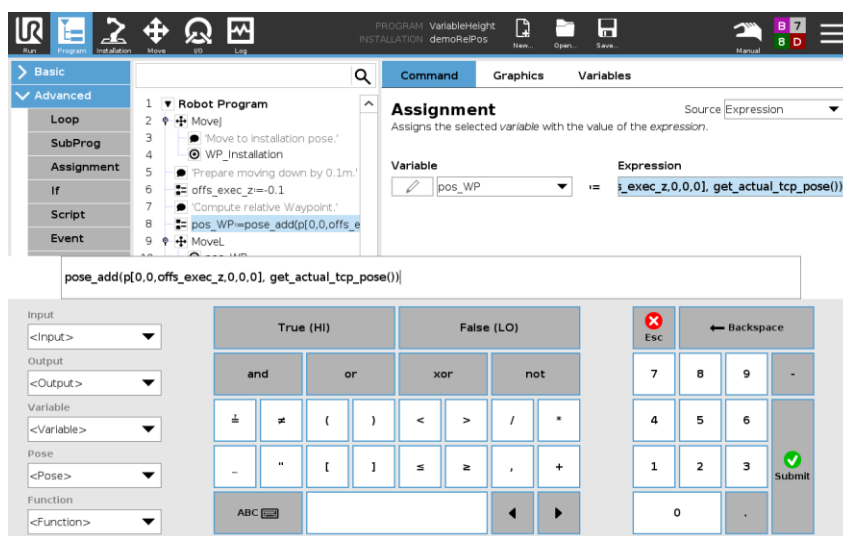


Abbildung 7

10

→ Basic → Waypoint

Fahre Wegpunkt an, dessen Position dem Wert der Variablen pos_WP entspricht.

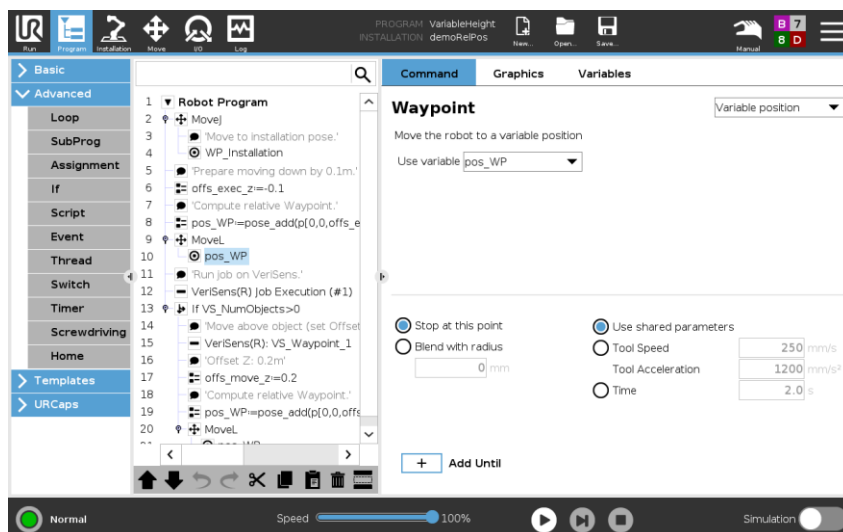
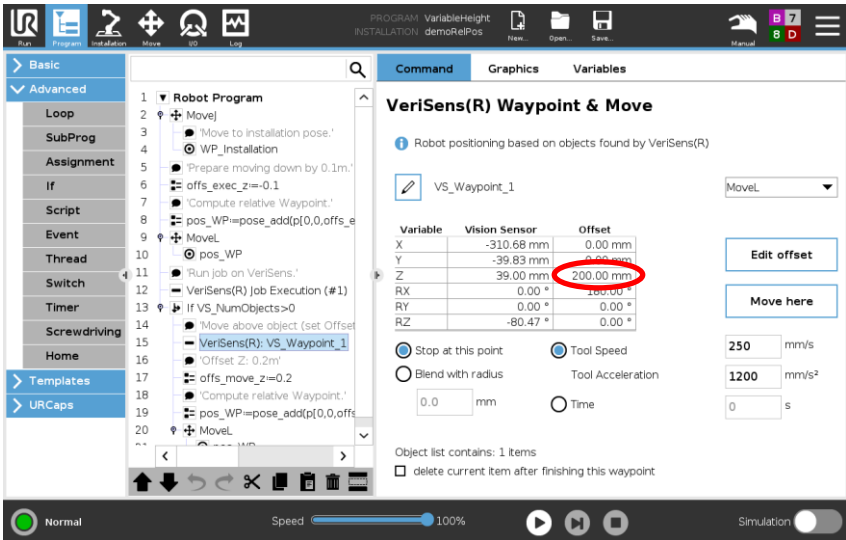


Abbildung 8

12

→ URCap → VeriSens(R) Job Execution

	<p>Führe ausgewählten Job auf <i>VeriSens</i>® aus.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>Hinweis</p> <p><i>VeriSens</i>® befindet sich in diesem Beispiel 100 mm tiefer als üblich. <i>VeriSens</i>® erhält über diese veränderte Bildaufnahmeposition keine Kenntnis und liefert als Z-Koordinate weiterhin die im Job in der <i>Application Suite</i> eingestellte Z-Höhe.</p> </div>
13	<p>→ Advanced → If</p> <p>Führe das folgende Unterprogramm nur aus, wenn mindestens ein Objekt gefunden wurde.</p>
15	<p>→ URCap → VeriSens(R) Waypoint & Move</p> <p>Fahre die von <i>VeriSens</i>® gelieferte Objektposition an, wobei diese manuell in Z-Richtung angepasst wird, um Kollisionen zu vermeiden.</p> <div style="text-align: center;">  </div> <p>Abbildung 9</p> <p>Diese manuelle Anpassung beträgt im Beispiel 200 mm. Damit fährt der Roboter 200 mm über die im Job in der <i>Application Suite</i> eingestellte Z-Position. Der Wert dieser Höhenanpassung ist so zu wählen, dass der Roboter in jedem Fall eine Position über dem Objekt anfährt.</p> <p>Insbesondere wenn <code>offs_exec_z > 0</code> ist, sollte diese manuelle Anpassung etwas grösser als der Wert von <code>offs_exec_z</code> sein.</p>
17	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>offs_move_z</code> auf denselben Wert, der zuvor im Knoten «<i>VeriSens</i>® Wegpunkt & Bewegen» (EN: «<i>VeriSens</i>® Waypoint & Move») (Zeile 13) manuell als Offset eingestellt wurde (hier im Beispiel 0,2 m).</p>
19	<p>→ Advanced → Assignment</p> <p>2. Höhenanpassung: setze Variable <code>pos_WP</code> auf einen berechneten Wert. Dieser wird mit <code>pose_add()</code> als Summe der folgenden zwei Größen gebildet:</p> <ul style="list-style-type: none"> - <code>p[0, 0, offs_exec_z - offs_move_z - 0.05, 0, 0, 0]</code> ... eine erzeugte relative Verschiebung in Z-Richtung, die sich aus drei Z-Komponenten zusammensetzt: <ul style="list-style-type: none"> o <code>offs_exec_z</code> ... berücksichtigt die vertikale Verschiebung während der Job Execution



	<ul style="list-style-type: none">o <code>-offs_move_z ...</code> kompensiert die vertikale Verschiebung des angefahren Wegpunkts über dem Objekto <code>-0.05 ...</code> manuelle Anpassung: z. B. für eine tiefer liegende Greifposition (hier im Beispiel Verschiebung um 50 mm in neg. Z-Richtung)- <code>get_actual_tcp_pose()</code> ... die aktuelle Position des Roboters
21	<p>→ Basic → Waypoint</p> <p>Fahre Wegpunkt an, dessen Position dem Wert der Variablen <code>pos_WP</code> entspricht.</p>

3.6 Programmbeispiel 2 – vier Schrauben abhängig von variabler Position

3.6.1 Beschreibung der Applikation

Im zweiten Beispiel (Abbildung 10) sollen abhängig von der durch *VeriSens*® gefundenen Position in der Mitte durch einen per Roboter geführten elektrischen Schrauber vier Schrauben in fester Position zur Mitte festgeschraubt werden. Z entspricht bei der Aufnahme der Soll-Höhe des Koordinatenabgleichs. Es wird dann für das «Holen» der Position ein höheres z angefahren und danach erfolgt eine Anpassung in z sowie in x und y für die Schraubposition jeder Ecke.

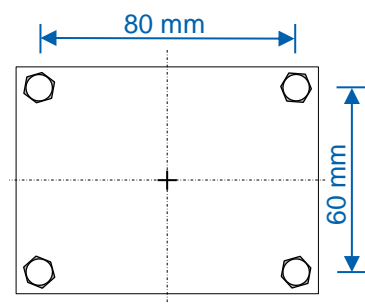


Abbildung 10

3.6.2 Anlegen der Variablen

Die folgenden Variablen werden im Programm verwendet und müssen daher vorher angelegt werden. Die zunächst festgelegten Werte werden im laufenden Programm angepasst.

`offs_corner_x`

X-Verschiebung gegenüber Objektmittle der Schraubposition einer Ecke

`offs_corner_y`

Y-Verschiebung gegenüber Objektmittle der Schraubposition einer Ecke

`offs_move_z`

Vertikale Verschiebung, die in *VeriSens*® Waypoint manuell eingestellt wurde

`pos_WP`

Berechnete Position eines Wegpunkts, der im Programm angefahren werden soll.

Syntax: `p[x, y, z, rotX, rotY, rotZ]`

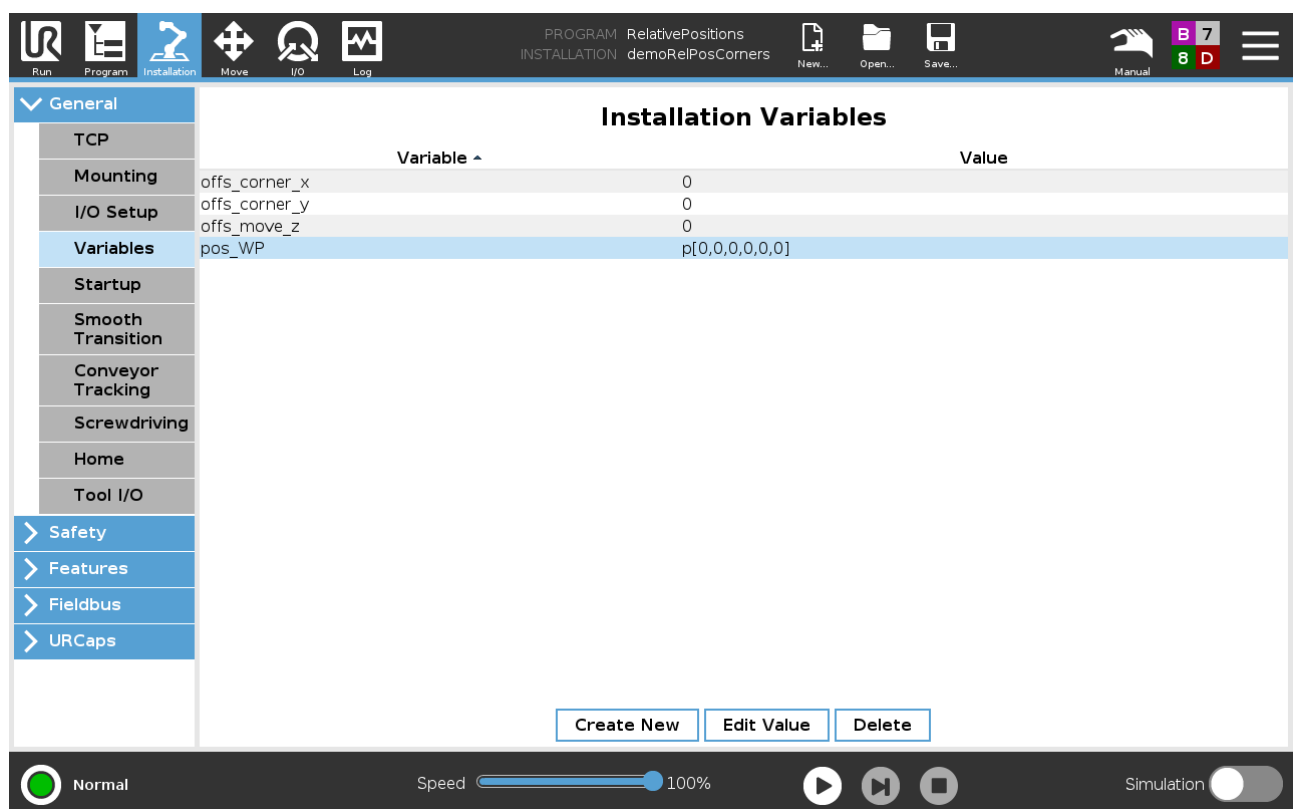


Abbildung 11

3.6.3 Erstellen des Programms

Mit der Roboterprogrammierung kann nun begonnen werden (Abbildung 11).

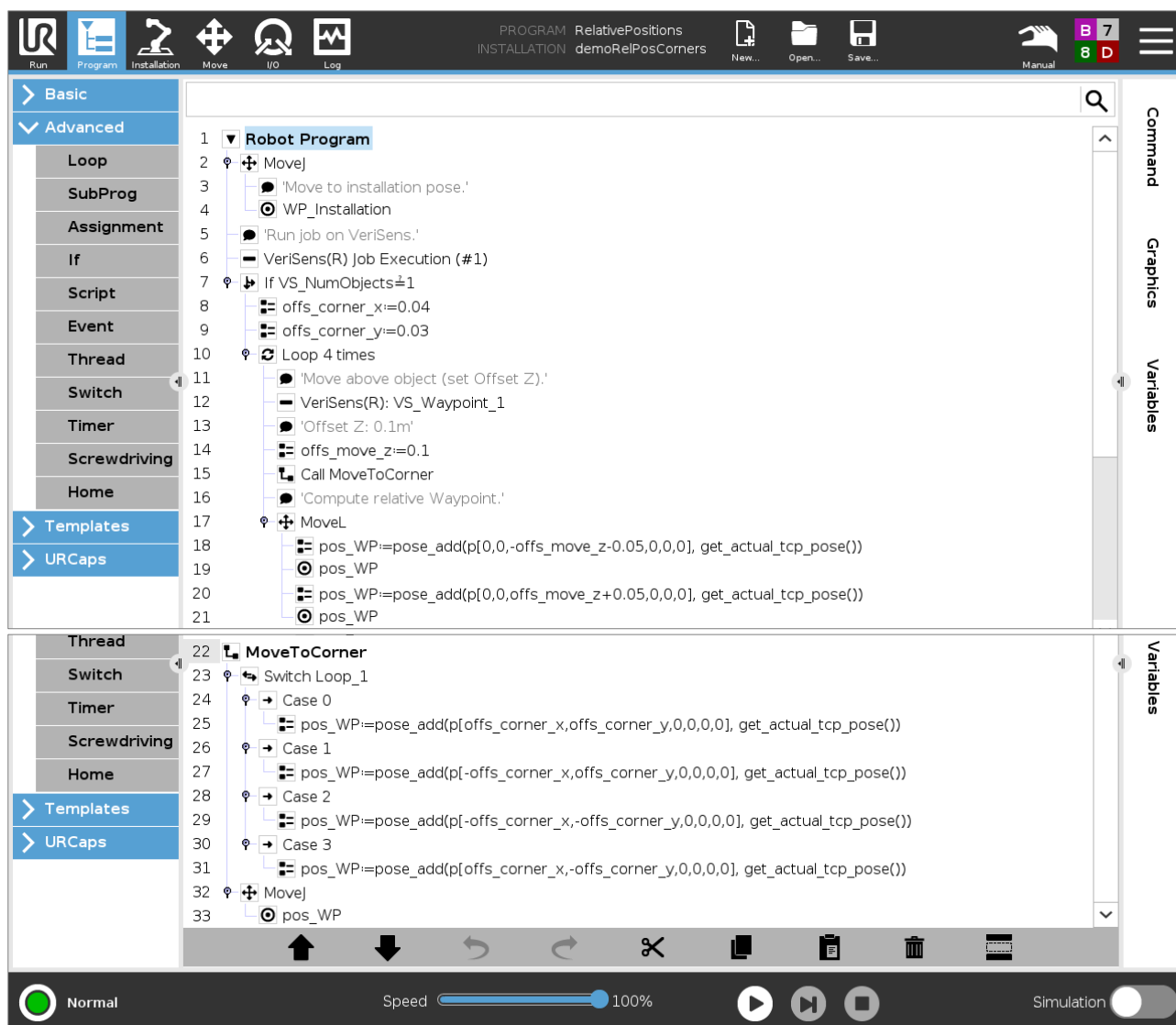
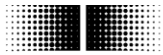
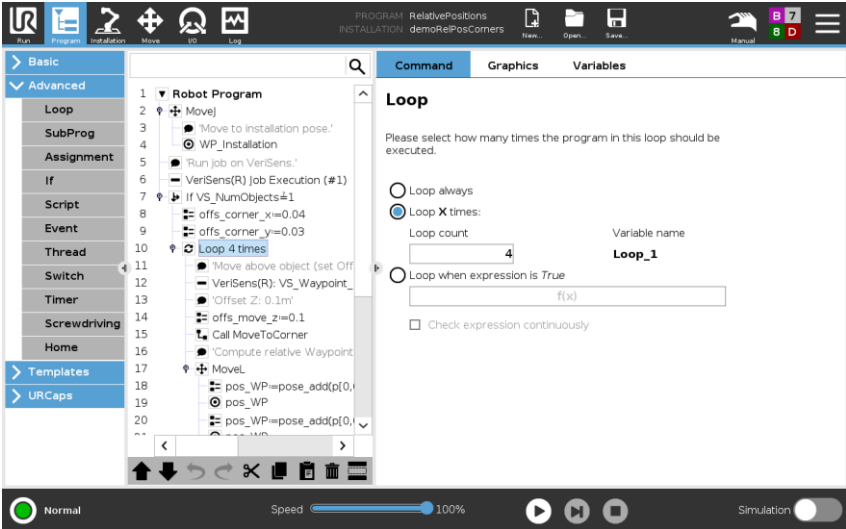


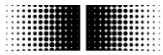
Abbildung 12

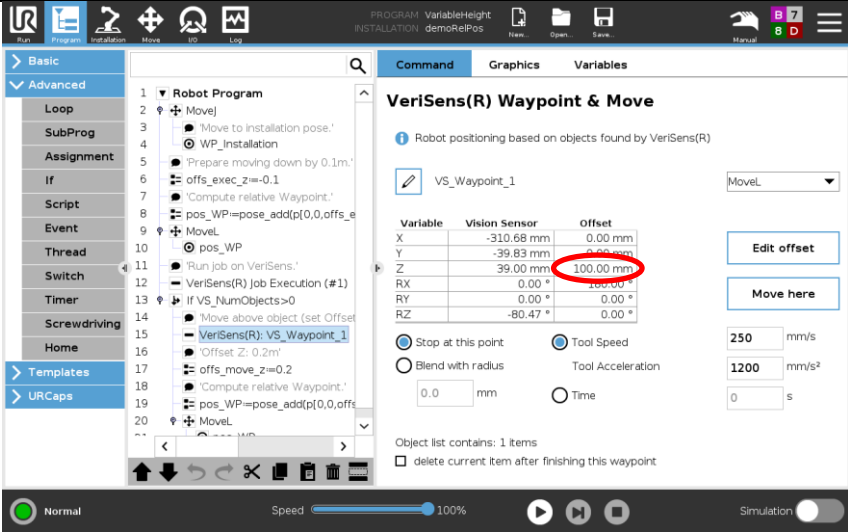
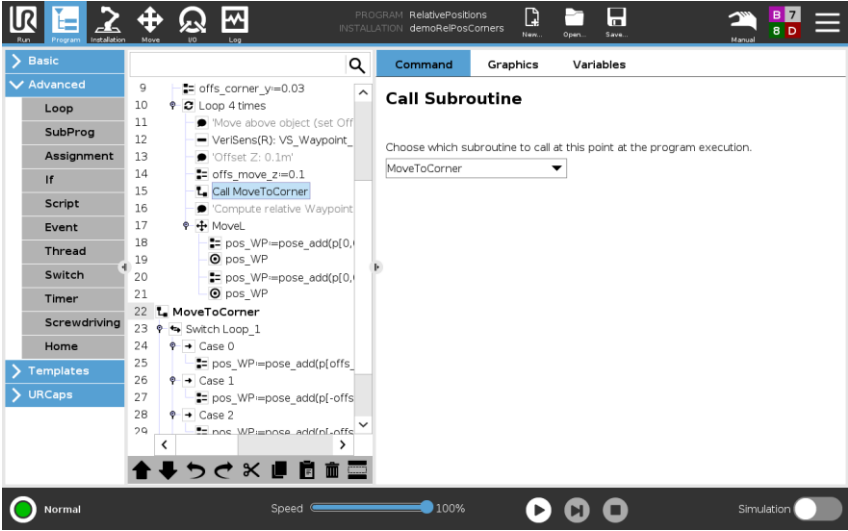
Programmbeschreibung

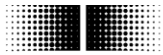
Zeile	Erläuterung
4	→ Basic → Waypoint Fahre Wegpunkt an, der entweder der Pose des Vision Sensors während der Installation entspricht oder einer Pose mit derselben Höhe z wie bei der Installation.
6	→ URCap → VeriSens(R) Job Execution Führe ausgewählten Job auf VeriSens® aus.
7	→ Advanced → If Führe das folgende Unterprogramm nur aus, wenn ein Objekt gefunden wurde.

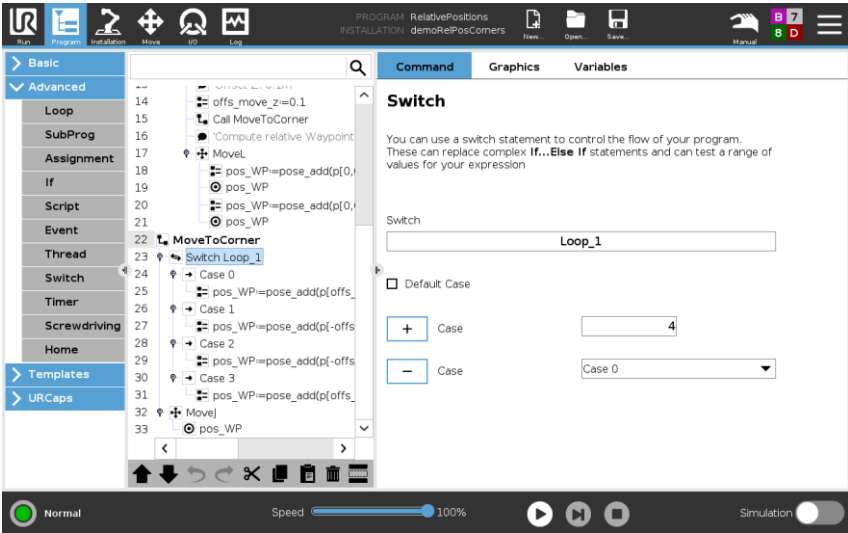


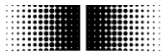
8	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>offs_corner_x</code> auf einen festen Wert.</p> <p><u>Beispiel:</u> <i>VeriSens</i>[®] liefert als Objektposition die Objektmittle. Der Wert <code>offs_corner_x</code> wird auf den halben Abstand zwischen den Schraubpositionen in X-Richtung eingestellt, da sich eine Schraubposition in diesem Abstand zur Mitte des Objekts befindet.</p>
9	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>offs_corner_y</code> auf einen festen Wert.</p> <p><u>Beispiel:</u> <i>VeriSens</i>[®] liefert als Objektposition die Objektmittle. Der Wert <code>offs_corner_y</code> wird auf den halben Abstand zwischen den Schraubpositionen in Y-Richtung eingestellt, da sich eine Schraubposition in diesem Abstand zur Mitte des Objekts befindet.</p>
10	<p>→ Advanced → Loop</p> <p>Führe eine Schleife ein, um die 4 Schraubpositionen nacheinander anzufahren.</p>  <p>Abbildung 13</p>
12	<p>→ URCap → VeriSens(R) Waypoint & Move</p> <p>Fahre die von <i>VeriSens</i>[®] gelieferte Objektposition an, wobei diese manuell in Z-Richtung angepasst wird, um Kollisionen zu vermeiden.</p>

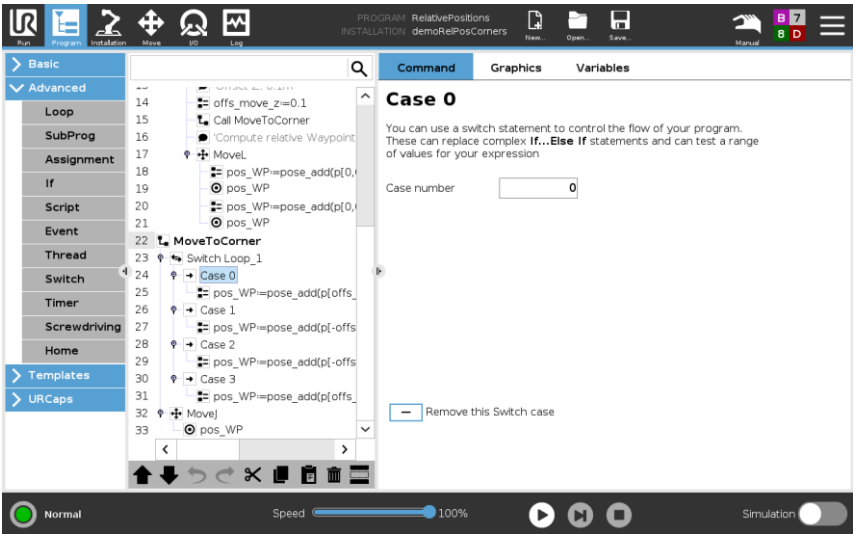


	<div></div> <p>Abbildung 14</p> <p>Diese manuelle Anpassung beträgt im Beispiel 100 mm. Damit fährt der Roboter 100 mm über die im Job in der <i>Application Suite</i> eingestellte Z-Position. Der Wert dieser Höhenanpassung ist so zu wählen, dass der Roboter in jedem Fall eine Position über dem Objekt anfährt.</p>
14	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>offs_move_z</code> auf denselben Wert, der zuvor im Knoten «<i>VeriSens®</i> Wegpunkt & Bewegen» (EN: «<i>VeriSens®</i> Waypoint & Move») (Zeile 12) manuell als Offset eingestellt wurde (hier im Beispiel 0,1 m).</p>
15	<p>→ Advanced → SubProg</p> <p>Führe ein das Unterprogramm „MoveToCorner“ aus (siehe dazu ab Zeile 22), wodurch in Abhängigkeit des Schleifenindex eine Position über einer der vier Schraubpositionen angefahren wird.</p> <div></div> <p>Abbildung 15</p>
18	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>pos_WP</code> auf einen berechneten Wert. Dieser wird mit <code>pose_add()</code> als Summe der folgenden zwei Größen gebildet:</p>



	<ul style="list-style-type: none">• <code>p[0, 0, -offs_move_z-0.05, 0, 0, 0]</code> ... eine erzeugte relative Verschiebung in Z-Richtung, die sich aus zwei Z-Komponenten zusammensetzt:<ul style="list-style-type: none">◦ <code>-offs_move_z</code> ... kompensiert die vertikale Verschiebung des angefahren Wegpunkts über dem Objekt◦ <code>-0.05</code> ... manuelle Anpassung: z. B. für eine tiefer liegende Schraubposition (hier im Beispiel Verschiebung um 50 mm in neg. Z-Richtung)• <code>get_actual_tcp_pose()</code> ... die aktuelle Position des Roboters
19	<p>→ Basic → Waypoint</p> <p>Fahre Wegpunkt an, dessen Position dem Wert der Variablen <code>pos_WP</code> entspricht.</p> <div>Hinweis Nun könnte der Schraubvorgang erfolgen.</div>
20	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>pos_WP</code> auf einen berechneten Wert. Damit wird die vertikale Positionsänderung aus Zeile 18 rückgängig gemacht.</p>
21	<p>→ Basic → Waypoint</p> <p>Fahre Wegpunkt an, dessen Position dem Wert der Variablen <code>pos_WP</code> entspricht.</p> <div>Hinweis Nun befindet sich der Roboter wieder in einer Distanz über dem Objekt.</div>
22	Beginn der Beschreibung des Unterprogramms „MoveToCorner“
23	<p>→ Advanced → Switch</p> <p>Hinzufügen von 4 Fällen (Case) innerhalb einer Switch-Anweisung. Damit kann jede Schraubposition separat innerhalb einer Case-Anweisung behandelt werden.</p>  <p>Abbildung 16</p>



24, 26, 28, 30	<p>Automatisch eingefügte Case-Anweisung, innerhalb derer weitere Knoten eingefügt werden können.</p>  <p>Abbildung 17</p>
25, 27, 29, 31	<p>→ Advanced → Assignment</p> <p>Setze Variable <code>pos_WP</code> auf einen berechneten Wert. Dieser wird mit <code>pose_add()</code> als Summe der folgenden zwei Größen gebildet:</p> <ul style="list-style-type: none">• <code>p[±offs_corner_x, ±offs_corner_y, 0, 0, 0, 0]</code> ... eine erzeugte relative Verschiebung in X- und Y-Richtung, um anschliessend (Zeile 33) den Roboter jeweils passend über eine Schraubposition bewegen zu können.• <code>get_actual_tcp_pose()</code> ... die aktuelle Position des Roboters
33	<p>→ Basic → Waypoint</p> <p>Fahre Wegpunkt an, dessen Position dem Wert der Variablen <code>pos_WP</code> entspricht.</p> <div><p>Hinweis</p><p>Nun befindet man sich abhängig vom jeweiligen Fall über einer Schraubposition. Die Programmausführung setzt mit dem Ende dieses Unterprogramms in Zeile 16 fort.</p></div>

4 Zusammenfassung/Spezialfälle

Man kann verschieden hohe Objekte prüfen bzw. deren Position erfassen und sogar Folgeaktionen damit steuern.

Die Methoden dafür sind vielfältig und abhängig von der Applikation. Während Variante 1 durch Einfachheit punktet und auch für Einsteiger schnell realisierbar ist, bietet Variante 2 alle Freiräume für erfahrene Programmierer.

5 Downloads

Ergänzend sei auf die Dokumentation zu *VeriSens*® Vision Sensoren, speziell der Abschnitt zu den Universal Robots, hingewiesen.

[Product Finder](#)

6 Support

Bei Fragen kontaktieren Sie bitte unser Technical & Application Support Center.

Worldwide

Baumer Optronic GmbH

Badstrasse 30 · DE-01454 Radeberg
Deutschland

Phone +49 3528 4386 845

support.verisens@baumer.com

7 Rechtliche Hinweise

Alle erwähnten Produkt- und Unternehmensnamen sind Marken oder eingetragene Marken der jeweiligen Inhaber.

Alle Rechte vorbehalten. Die ganze oder auszugsweise Vervielfältigung dieses Dokuments ist nur mit vorheriger schriftlicher Genehmigung der Baumer Optronic GmbH zulässig.

Änderungen im Sinne des technischen Fortschritts sowie eventuelle Irrtümer vorbehalten.

Baumer Group

Die Baumer Group ist einer der international führenden Hersteller von Sensoren, Drehgebern, Messinstrumenten und Komponenten für die automatisierte Bildverarbeitung. Baumer verbindet innovative Technik und kundenorientierten Service zu intelligenten Lösungen für die Fabrik- und Prozessautomation und bietet dafür eine einzigartige Produkt- und Technologiebreite. Das Familienunternehmen ist mit rund 2.700 Mitarbeitern und Produktionswerken, Vertriebsniederlassungen und Vertretungen in 39 Niederlassungen und 19 Ländern immer nahe beim Kunden. Mit weltweit gleichbleibend hohen Qualitätsstandards und einer grossen Innovationskraft verschafft Baumer seinen Kunden aus zahlreichen Branchen entscheidende Vorteile und messbaren Mehrwert. Weitere Informationen im Internet unter www.baumer.com.

**Baumer Optronic GmbH**

Badstrasse 30 · DE-01454 Radeberg
Phone +49 3528 4386 0 · Fax +49 3528 4386 86
sales@baumeroptronic.com · www.baumer.com