Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

KRL	Kuka Roboter Language
TCP	Tool Center Point
VT	VisionTech-Befehl
INT	Integer-Variable (Ganzzahl-Variable)
RGB	Rot-Gelb-Blau-Farben

1 Einleitung

Für die Sicherstellung der Produktion ganzer Industrien werden Roboter immer wichtiger. Nicht nur in Bereichen wie der Automobilbranche oder der Forschung, sondern auch in der Medizintechnik werden bereits Roboter verwendet, deren Präzision und Einsatzfähigkeit die eines Menschen übertrifft. Kleine Operationen sind hier schon seit einigen Jahren an der Tagesordnung. Doch der Roboter ist oft nicht nur die einzige Komponente die Arbeit in gleichbleibend hoher Qualität zu vollbringen. Ist dieser für eine flexible Tätigkeit vorgesehen bzw. soll er auf verschiedene Einflüsse reagieren können, so kommt oft eine Kamera zum Einsatz, welche in Echtzeit Daten mit dem Roboter austauscht und diesen somit unabhängiger als ohne solcher Sensoren macht. In einem automatisierten Lagerregal werden über eine Schrifterkennungssoftware zum Beispiel Barcodes ausgelesen und zugeordnet werden. In großen Unternehmen ist es mit einer solchen Schrifterkennungssoftware möglich eine chaotische Lagerhaltung einzuführen. Damit können Teile nicht nur aufgrund von Reihenfolgen in der Fertigung oder Einsatzbereich eingelagert werden sondern auch nach Größe sortiert werden, um somit Platz und Geld zu sparen. Die Teile werden vom Roboter eingelagert und wenn es nötig ist wird der Barcode aus dem System herausgelesen. Danach werden die Teile vom Roboter abgeholt. An der Hochschule Mittweida gibt es die Möglichkeit genau diese Kommunikation zwischen Kamera und Roboter herzustellen und die Funktionsweise zu lernen.

Die vorliegende Arbeit befasst sich mit dem Thema der kamerabasierten Objekt– und Mustererkennung. Als Visionsystem kommt hier das der Firma Kuka mit den dazugehörigen Programmen WorkVisual und VisionTech zum Einsatz. Die Firma Kuka KUKA Roboter GmbH mit Sitz in Augsburg zählt mit seinen ca. 12000 Mitarbeitern weltweit zu den führenden Anbietern für intelligente Automatisierungslösungen[1]. Dabei liefert KUKA neben dem Roboter auch die nötigen Komponenten, die Zelle sowie die fertige und vollständige Anlage. Die Hochschule Mittweida pflegt seit Jahren den Kontakt zur bayerischen Firma und hat im Jahr 2015 ebenfalls neue Trainingszellen mit Visionsystemen erworben. Diese kommen im folgenden Projekt zum Einsatz und werden in den jeweiligen Abschnitten genauer beschrieben.

1.1 Aufgabenstellung und Zielsetzung

Aufgabe ist es eine Experimentierzelle von Kuka mit einer Kamera so zu installieren, dass verschiedene Untersuchungen damit gemacht werden können. Im Laufe der Arbeit wird auf verschiedene Punkte eingegangen. Zum einen werden die unterschiedlichen mitgelieferten Programme behandelt. Diese sind das in der Kukasteuerung enthaltene VisionTech und das am Partnercomputer zu bedienende Programm WorkVisual. Beide arbeiten Hand in Hand und sind für die Mustererkennung unabdingbar.

Außerdem wird ein Programm geschrieben und erklärt, wie die Kamera ein zuvor eingelerntes Muster erkennt, der Roboter dieses anfährt und wegbringt. Neben dem Punkt der Mustererkennung wird noch eine Barcodeerkennung abgearbeitet. Dieser ist auf dem zu findenden Muster aufgeklebt und wird im Programm untersucht und ausgegeben. Aufgrund verschiedener Probleme, die während der Arbeit am Roboter auftraten und durch die Komplexität der Software, wird sich ein Abschnitt mit der Behandlung mancher Probleme und mit der Zuordnung oft benutzter Dateipfade befassen.

Oberstes Ziel dieses Projektes ist es mit dieser Zelle Objekterkennungen durchzuführen. Die Arbeit dient als Anleitung und soll Studenten helfen in Praktika oder Experimenten zu verstehen, wie eine Bildverarbeitung funktioniert und welche Möglichkeiten sich in Verbindung mit einem Roboter ergeben.

2 Voreinstellungen

In diesem Abschnitt werden die Voraussetzungen für die Bearbeitungen geklärt. Hierbei geht es um die Ausgangssituation der Hardware, also der Kamera und die des Roboters. Die Programme WorkVisual und VisionTech werden ebenfalls erklärt wie auch die Kommunikation zwischen Kamera und Roboter. Bereits in diesem Abschnitt werden Voraussetzungen geschaffen, welche für die Bilderkennung äußerst wichtig und somit entsprechend genau durchzuführen sind.

2.1 Stand der Technik

Die Kamera kommt ebenfalls von der Firma Kuka. Es handelt sich hier um eine stationäre 2D-Kamera. Im Gegensatz zu einer robotergeführten Kamera hängt diese direkt über dem zu fotografierenden Objekt und ist fest mit einer Halterung verbunden. 2D- Steuerung bedeutet, dass nur Kanten und Musterungen wahrgenommen werden können. Es können keine dreidimensionalen Berechnungen erstellt werden, sondern nur über einen Farbverlaufsschema Positionen und Orientierungen dargestellt werden.

Bild 1: Kamera

Der Farbverlauf wird dabei über ein Histogramm berechnet. Es wird dabei jedoch nicht in RGB-Farben unterschieden, sondern es wird ein Grauwertverlauf dargestellt. Die im Labor verwendete Kamera ist also keine Farbkamera. Diese erkennt Kanten anhand von Schattierungen und gibt die Ergebnisse in deren Abhängigkeit mit einer Güte in Prozent aus. Daher muss darauf geachtet werden, dass eine ausreichende Beleuchtung vorhanden ist. Ist das nicht der Fall, oder hebt sich das Teil zu schlecht vom Untergrund ab, kann das Ergebnis erheblich verfälscht werden.

Ein weiterer Punkt in der Beleuchtung ist das Problem mit wechselnden Lichtverhältnissen. Es gibt die Möglichkeit festeingestellte Kontrastwerte und Schwellwerte einzurichten und auf deren Basis eine Berechnung des Fotos durchzuführen. Gibt es am Versuchsplatz z.B. vor einem Fenster zur Sonnenseite wechselnde Lichtverhältnisse, so kommt es dadurch zur starken Beeinflussung der Kontrastwerte. Damit sind diese für die Berechnung meistens nicht zu gebrauchen und es ist deswegen von einer solchen fixen Einstellung der Werte abzuraten. Weiterhin muss beachtet werden, dass es bei verschiedenen Einfallswinkeln zur Schattenbildung kommen kann. Da die Kamera anhand des Grauwertverlaufs Kanten und Objekte bestimmt, passiert es also hin und wieder, dass Objekte erkannt werden, die es entweder nicht gibt oder die an einem anderen Ort liegen.

Nun zum Roboter. Dieser ist ein Sechs-Achsenroboter mit einer KRC4 Steuerung. Er stammt aus dem Hause KUKA und ist an der Hochschule ein neues Modell mit dem Namen Agilus. Vorarbeiten wurden seitens KUKA und der Professoren der Hochschule geleistet, indem Achsen eingerichtet, justiert und vermessen wurden. Falls dies nochmal durchgeführt werden muss, findet man das Menü im VisionTech mit einem Klick auf das Robotersymbol  in der linken oberen Ecke des Menüs (siehe Bild 3). Dabei kann man zwischen Justieren, Vermessen und anderen Systemeinstellungen wählen. Anweisungen werden gegeben und müssen befolgt werden. Weiterhin wurde die Steuerung des Roboters mit der Hardware verbunden und die Kommunikation über eine Ethernetschnittstelle eingerichtet. Somit wurden grundlegende Arbeiten vorab geleistet und der Roboter war einsatzbereit. Die Roboterzelle ist eine „normale“ Trainingszelle von KUKA. In dieser sind enthalten: ein Stiftmagazin, um Konturen zu zeichnen und/oder Positionen zu zeigen. Weiterhin ein Ablagetisch, auf dem gleichzeitig Konturen aufgedruckt sind, die in Beispielprogrammen abgefahren werden können, um sich so mit der Programmiermethodik in der KRL

vertraut zu machen. Nachfolgendes Bild soll einen Überblick geben, wo sich welche Teile befinden.

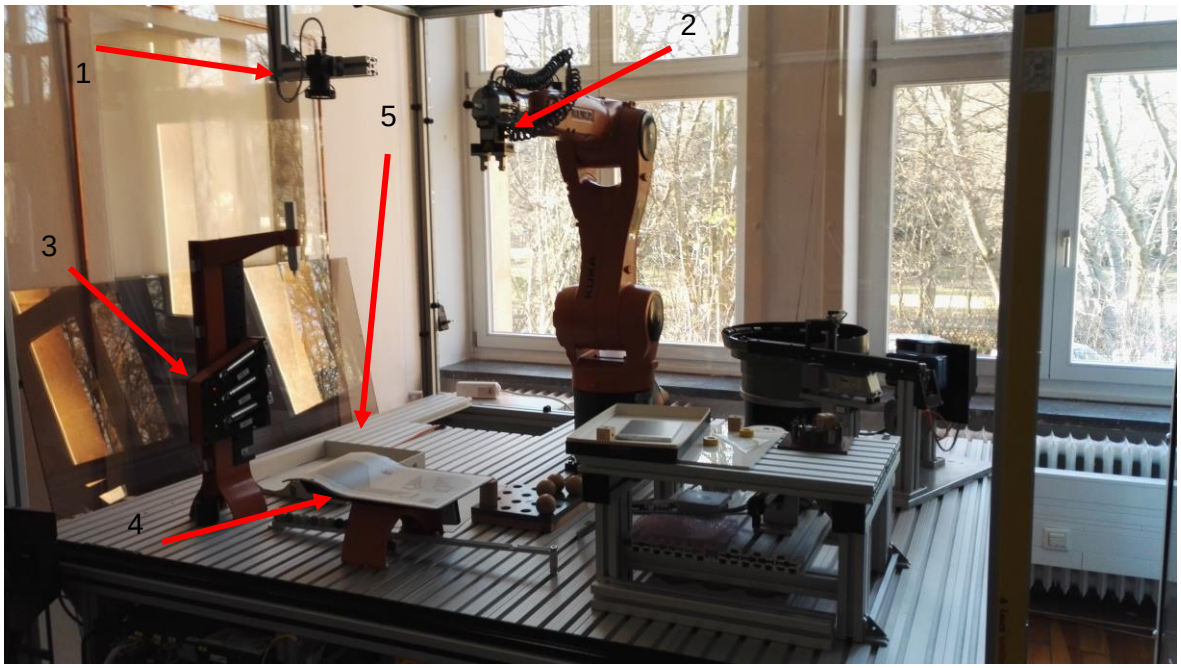

Bild 2: Experimentierzelle

1……Kamera mit Halterung

2……Zentralhand (TCP, Werkzeugwechsler)

3……Stiftmagazin

4……Ablagetisch

5……Bildbereich

Die Kamera wurde an einem dafür gefertigten Winkel angebracht. Sie wurde in T-Nuten an den Winkel geschraubt und kann somit in zwei Richtungen bewegt werden, wodurch auch auf den Bildbereich Einfluss genommen werden kann. Man sollte jedoch darauf achten, dass die Kamera nicht zu weit vom zu untersuchenden Bereich entfernt ist, da sonst Fehler auftreten und das Programm nicht mehr zuverlässig genug arbeitet. Im Labor sollte das Problem, wenn überhaupt, nur sehr selten auftreten. Eine absolute Aussage zum Abstand der Kamera kann hierbei nicht gegeben werden, da es auf den Kontrast des Bildes, sowie auf die Größe des zu untersuchenden Feldes ankommt.

Der TCP befindet sich am Greifer zwischen den beiden Greifbacken. Weiterhin ist an der Zentralhand ein Werkzeugwechsler vorhanden. Es kann somit zwischen einem Greifer und einem Sauger unterschieden werden. Damit können nicht nur winklige Objekte wie Würfel gegriffen,

sondern auch Kugeln oder ebene Objekte angesaugt und bewegt werden.

Im Stiftmagazin ist Platz für drei Stifte. Zum Einmessen der Basis sollten diese Stifte benutzt werden, da damit die größte Genauigkeit erzielt wird. Mit Beispielprogrammen, welche bereits am Arbeitsplatz existieren, können die Stifte auch vom Roboter aus dem Magazin herausgeholt werden. Der Ablagetisch ist selbsterklärend und zum Bildbereich wurde bereits das Wesentliche geschrieben.

Nun zur Bildverarbeitung. Die Bildverarbeitung geschieht mittels zweier Programme. Dabei wird in zwei Teile unterschieden. Zum einen ist es der Teil, der der Kamera angelernt werden muss, wie z.B. „das zu findende Muster einlernen“. Dies geschieht im WorkVisual und wird am Partnercomputer durchgeführt. Zum anderen sind es die Einstellungen, welche auf der Robotersteuerung verarbeitet werden. Hierzu zählen u.a. die Belichtungszeiten der Kamera und die Festlegung einer Kalibrierbasis. Diese Arbeiten werden mit der Software VisionTech realisiert. Im VisionTech werden Daten eingegeben und gespeichert, welche später als Task wieder aufgerufen werden. Diese Tasks beinhalten sogenannte Toolblocks, welche wiederum im WorkVisual erstellt und per Dateiübertragung an die KRC4 übermittelt werden. Im Roboterprogramm erfolgt dann über spezielle VisionTech-Befehle der Aufruf dieser Tasks und somit die Abarbeitung des benötigten Toolblocks. Über den genauen Ablauf wird in den entsprechenden Abschnitten geschrieben. Da unter Umständen sehr oft zwischen den Computern und Steuerungen gewechselt werden muss gibt es einen eigenen Abschnitt, welcher darüber aufklärt, wo Bilder, Dateien und Ordner zu finden sind.

Die Kommunikation geschieht im Großen und Ganzen über eine TCP/IP-Schnittstelle. Das Internetprotokoll ist für den Datenaustausch zwischen den beiden Medien zuständig. Nicht als Kommunikationsart, aber als Darstellungsvariante ist es ebenfalls möglich, das User-Interface der Robotersteuerung auf dem PC darzustellen. Hierzu kommt das Programm TeamViewer zum Einsatz. Darin stellt man eine Verbindung am PC her, indem man das Gerät auswählt und dieses mit Hilfe eines Kennwortes koppelt. Ab jetzt ist es ebenso möglich direkt vom PC zu programmieren. Zu beachten ist, dass z.B. bei Bildeinstellungen die Bilder und Strukturen der Fotos auf dem Bedienpanel besser zu erkennen und manche Eingaben schneller zu tätigen sind. Somit sollte man sich bei Vorabeinstellungen an der Anzeige auf dem Panel orientieren.

2.2 Kameraeinstellungen

Bevor mit der eigentlichen Arbeit begonnen werden kann müssen Einstellungen an der Kamera vorgenommen werden. Die Bildsättigung sowie der Kontrast können am Einstellrad an der Kamera verstellt werden. Zuerst muss man ein Livebild schalten, indem man auf das Robotersymbol klickt und im Punkt VisionTech ein Livebild schaltet.

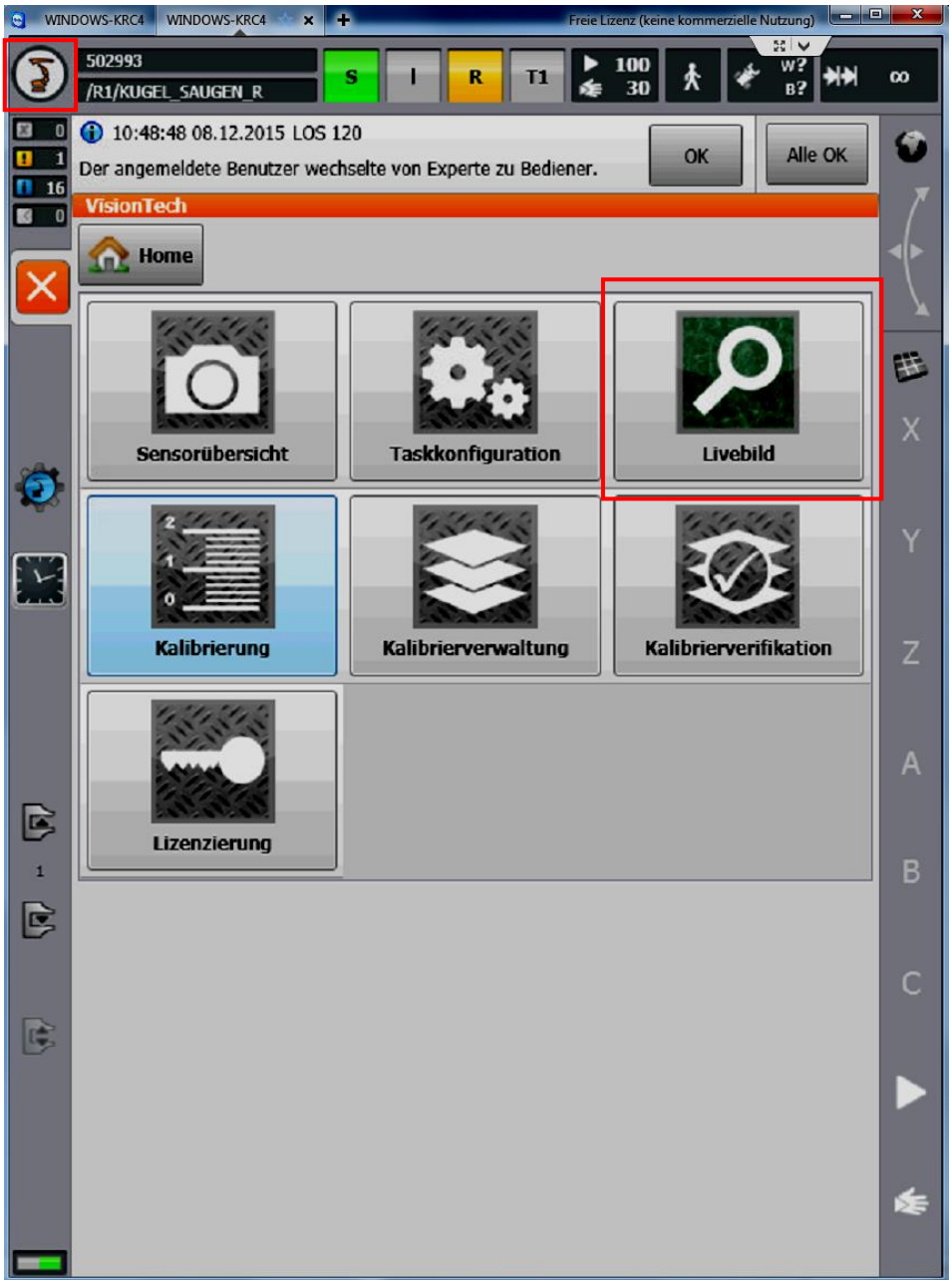

Bild 3: Livebild

Hier muss nun ein Sensor ausgewählt werden und nach einigen Sekunden sieht man den aktuellen Bildbereich der Kamera. An der Stelle kann man auch diesen Bildbereich nochmals kontrollieren. Möglicherweise muss die Position der Kamera verändert werden. Zu beachten ist, dass die Kameraeinstellungen erst vorgenommen werden sollten, wenn die Kamera richtig platziert ist. Mit einem Blick auf das Bild und durch Drehen am Einstellrad justiert man solange, bis die Schärfe hinreichend gut ist. Eine absolute Aussage zur Einstellung kann nicht getroffen werden. Die Belichtungsstärke lässt sich auch im am PC einstellen wenn die Kalibrierung der Base geschieht. Die Base beschreibt die Ausrichtung des Koordinatensystems.

2.3 Ethernet/KRL-Einstellungen

Es kommt ein Internetprotokoll zum Einsatz, um die Kommunikation zwischen den Komponenten zu gewährleisten. Hierzu ist es notwendig, dass einzelne Flags abgefragt werden. Diese Flags sind sozusagen Statusindikatoren und haben die Aufgabe zu signalisieren, ob Daten empfangen wurden oder ob darauf gewartet wird, dass welche geschickt werden. Weiterhin wird ein Interrupt eingefügt wenn diese Flags aufgerufen werden. Dieser Interrupt wird ausgeführt, wenn die Bildbearbeitung beendet ist und ruft das Ergebnis ab. Die Einstellungen der Flags sind zwar vorhanden, jedoch ist es notwendig diese zu definieren und später die definierten im Programm aufzurufen. Folgende Bilder sollen den Weg in die config-Datei aufzeigen, um die Einstellmöglichkeit schneller zu finden. Hierzu muss man sich zuerst als Experte anmelden, um Zugriff auf diesen Bereich zu haben. Danach sucht man sich im Menü das Laufwerk C und danach auf den Ordner KRC.

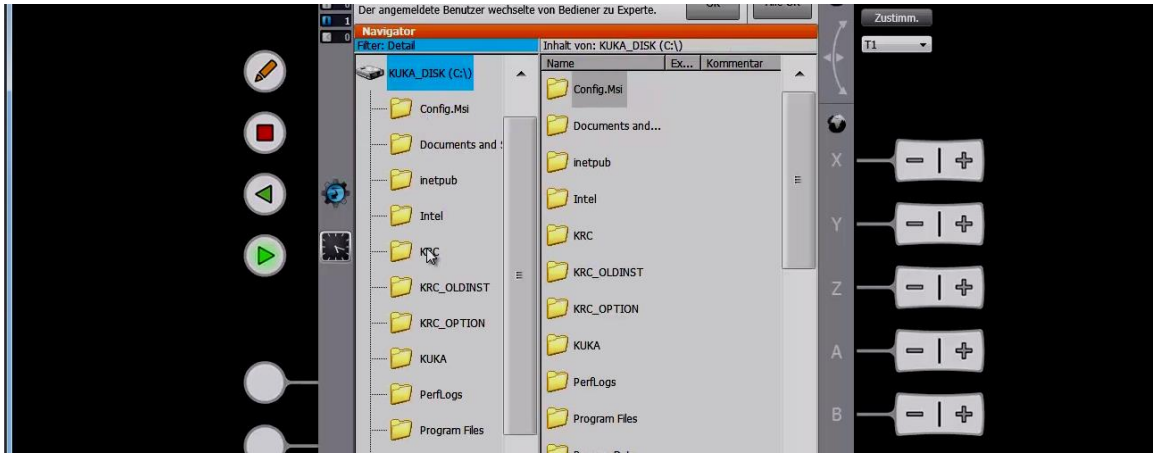

Bild 4: Etherneteinstellung Laufwerk C

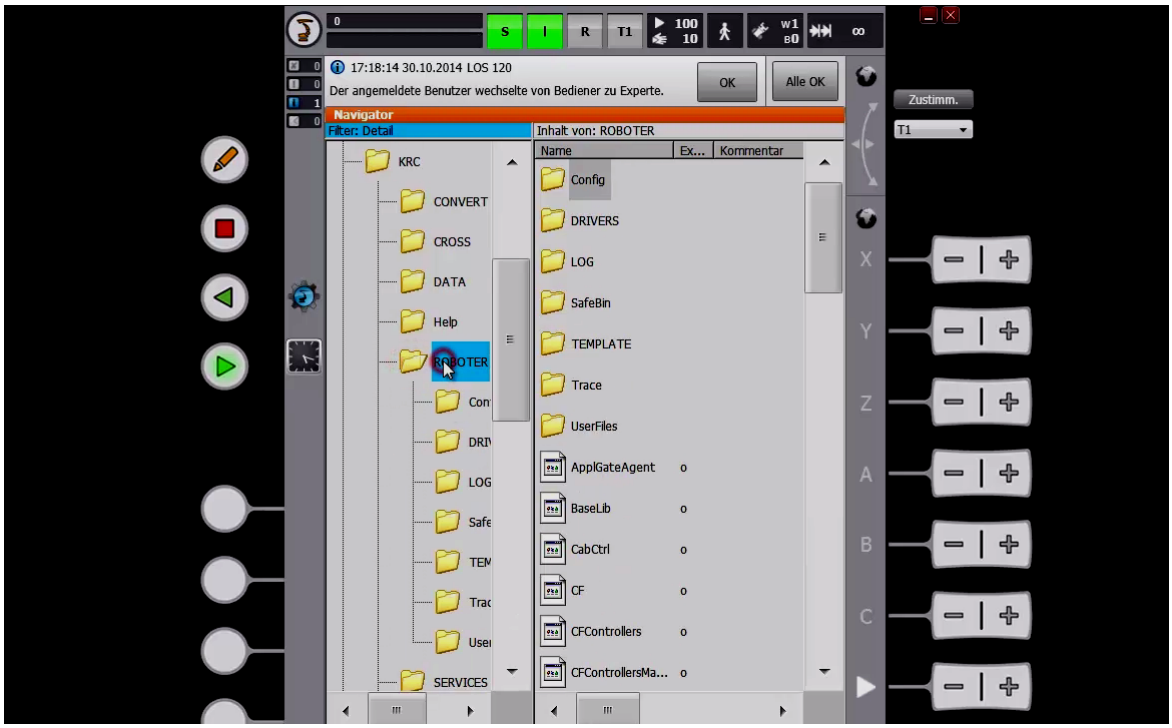

Bild 5: Etherneteinstellung Roboter

Im Ordner Roboter befinden sich der Config-Ordner und darin die U-ser-Einstellungen.

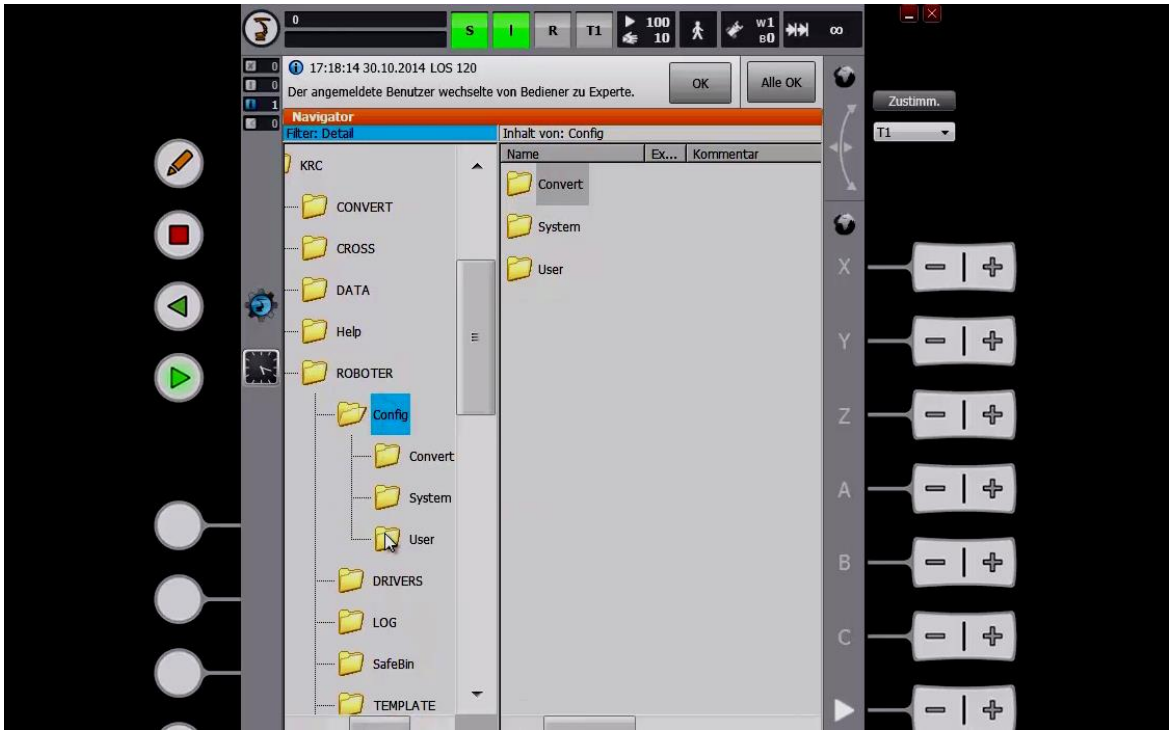

Bild 6: Etherneteinstellung Config

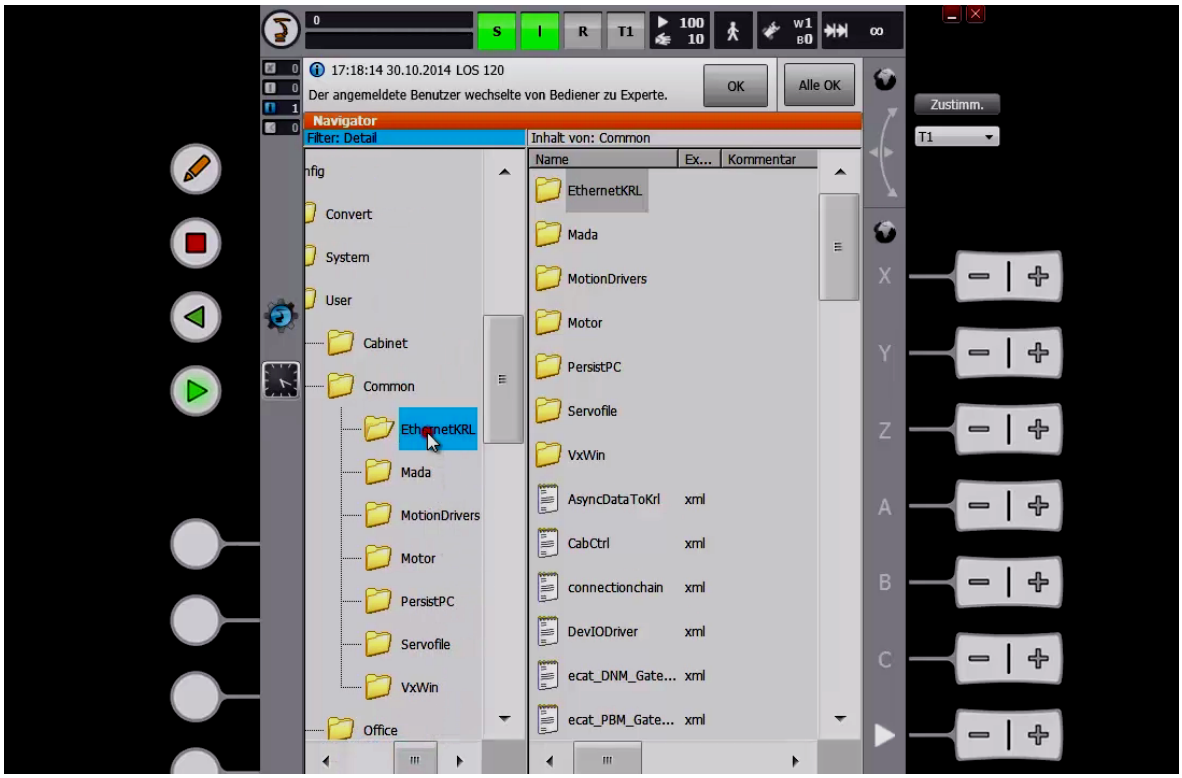

Bild 7: Etherneteinstellung Ethernet KRL

Im EthernetKRL-Ordner sucht man sich die VisionTechConfig.xml und öffnet diese.

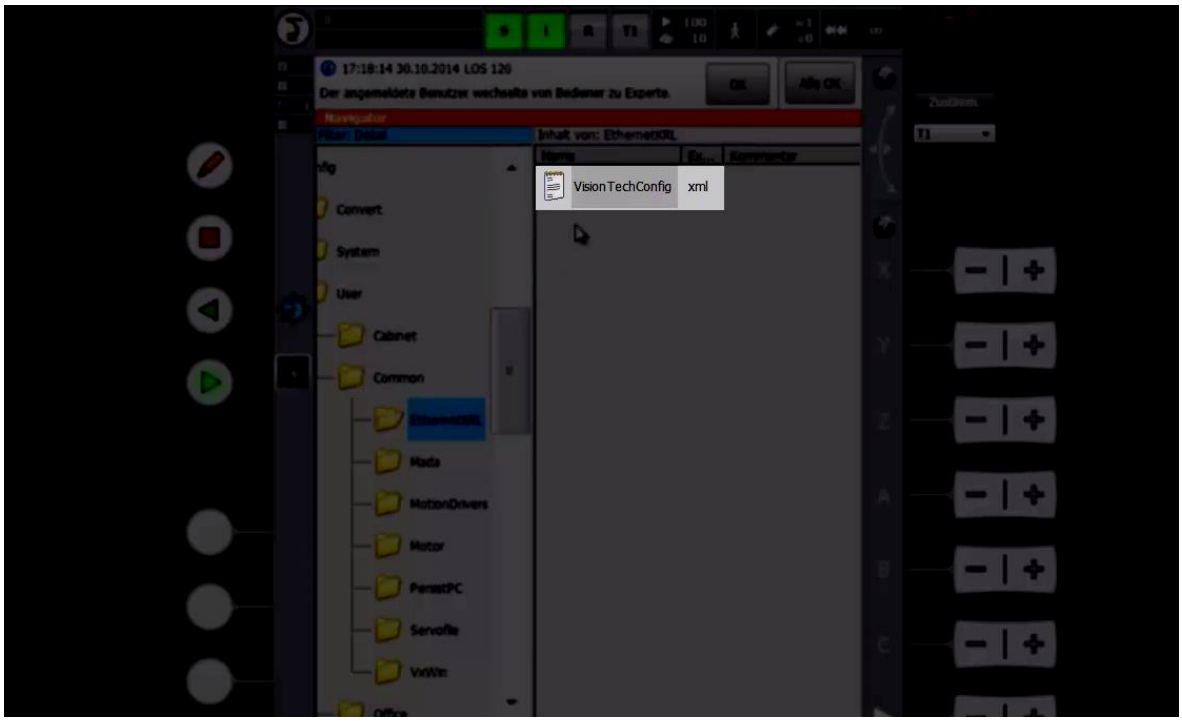

Bild 8: Etherneteinstellung VisionTechConfig.xml

Hier sind verschiedene Variablen definiert. Da die Flags mit verantwortlich für das Ergebnis sind, befinden sich diese im Abschnitt EndOfResult. Dieser Abschnitt ist der letzte Schritt in der Abarbeitung der Prozedur. Erst wenn dieser ausgeführt wurde, wird das eingestellte Flag gesetzt.

Bild 9: Etherneteinstellung EndOfResult

Um das Flag aufzurufen vergibt man diesem einen Namen. Hier und im Beispielprogramm wurde die 998 verwendet. Dazu löscht man den bisherigen Text „Choose_Your_Flag“ und fügt an der Stelle „998“ ein. Danach werden die Änderungen gespeichert und die Einstellungen für die Verbindung sind beendet.

Bild 10: Etherneteinstellung Flag

Jetzt müssen die Treiber der Ein-und Ausgänge rekonfiguriert werden. Das geschieht in folgendem Ordner: **Konfiguration→Ein-/Ausgänge→E/A Treiber→ Rekonfigurieren**.

3 Erkennung von Bauteilen

In diesem Kapitel dreht sich alles um den Programmaufbau und Ablauf der Mustererkennung.
Zuerst wird die Kalibrierung durchgeführt. Da beim Aufbau des Programms Probleme entstanden wird hier darauf hingewiesen, die Kalibrierung mit äußerster Sorgfalt durchzuführen, weil sich bereits kleine Fehler später im Programm bemerkbar machen werden. Nach der Kalibrierung kommt es zur Task- und Toolblockerstellung im WorkVisual. Es werden dabei die Toolblocks für die einzelnen Erkennungsarten besprochen und auf bemerkenswerte Einstellungen hingewiesen. Im letzten Abschnitt geht es um die Programmierung der Routine. Es werden dabei alle Einzelschritte genau erläutert und teilweise mit Grafiken untermauert.

3.1 Kalibrierung der Kamera

Bevor es zur eigentlichen Programmierung kommen kann müssen vorher bestimmte Voraussetzungen erfüllt sein. Hierzu gehören das Einstellen der Kamera sowie das Trainieren des Musters, welches gefunden werden soll. In diesem Abschnitt handelt es sich um die Abarbeitung der notwendigen Schritte und deren Bedeutung für das Projekt.

- Kamera in Ausgangsposition befestigen
- Belichtungszeit im Livebild einstellen
- Kontrast mit dem Einstellring an der Kamera verändern
- Kamera im VisionTech kalibrieren
- Kalibrierkörper auswählen
- Kalibrierassistent auswählen und Bild aufnehmen →**Kalibrieren**
- Kalibrierebene, Basis und Name vergeben
- neue Basis anlegen → auf Ausrichtung des Fiducial (siehe Seite 26) achten!

Kameraeinstellungen

Um die optimale Einstellung der Kamera zu finden, muss diese erste in ihre Ausgangsposition gebracht werden. Im Labor wurde dazu ein Winkel dafür angefertigt, um die Kamera einerseits stabil zu positionieren und andererseits die Lage auch je nach Auftrag oder Projekt zu verändern. Ist das geschehen, hängt die Kamera mittig und wie gewünscht am Platz, muss dann die Belichtungszeit eingestellt werden. Hierzu ist es nötig zuerst über das Robotersymbol im VisionTech ein Livebild zu schalten und dieses zu starten. Nach einiger Zeit (es kann auch mal 20 Sekunden dauern) öffnet sich ein Fenster, in dem man sich zwischen 0 und 200 ms die optimale Belichtungszeit auswählen kann. Da im Labor die Lichtverhältnisse durchaus gut sind, jedoch auch im Sommer durch direkte Sonneneinstrahlung unterschiedlich sein könnten, wurde eine Belichtungszeit von ca. 40 ms (je nach Situation) ausgewählt.

Bild 11: Belichtungszeit

Zur Unterstützung für die Kamera bzw. zur besseren Ausbeute von brauchbaren Resultaten wurde ein LED-Kranz um die Kamera befestigt, sodass noch eine zusätzliche Lichtquelle zu Hilfe genommen werden kann. Es ist weiterhin möglich den Kontrast der Kamera zu verändern. Hierzu muss der LED-Kranz entfernt und die darunter liegende Abdeckung abgeschraubt werden. Veränderungen kann man wieder mit Hilfe des Livebildes verfolgen. Nach dem Setzen der aktuellen Einstellungen ist die Kamera vorbereitet.

Nachdem die ersten Einstellungen vorgenommen wurden muss nun die Kamera kalibriert werden. Das muss geschehen, um einerseits der Kamera einen mittleren Fehler einzugestehen. Zum anderen werden grobe Ungenauigkeiten herausgefiltert.

Mit einem Klick auf das Robotersymbol und anschließend auf Vision-Tech, wo man auf **Kalibrierung** klickt, öffnet sich ein erstes Auswahlfenster. Hier muss zuerst die Kalibrierplatte eingestellt und der Sensor aktiviert werden. Im Projekt wurde die 3,175mm Platte benutzt. Mit einem Tastendruck auf das darunter stehende Bild wird ein Foto gemacht und der Sensor ist somit aktiviert.

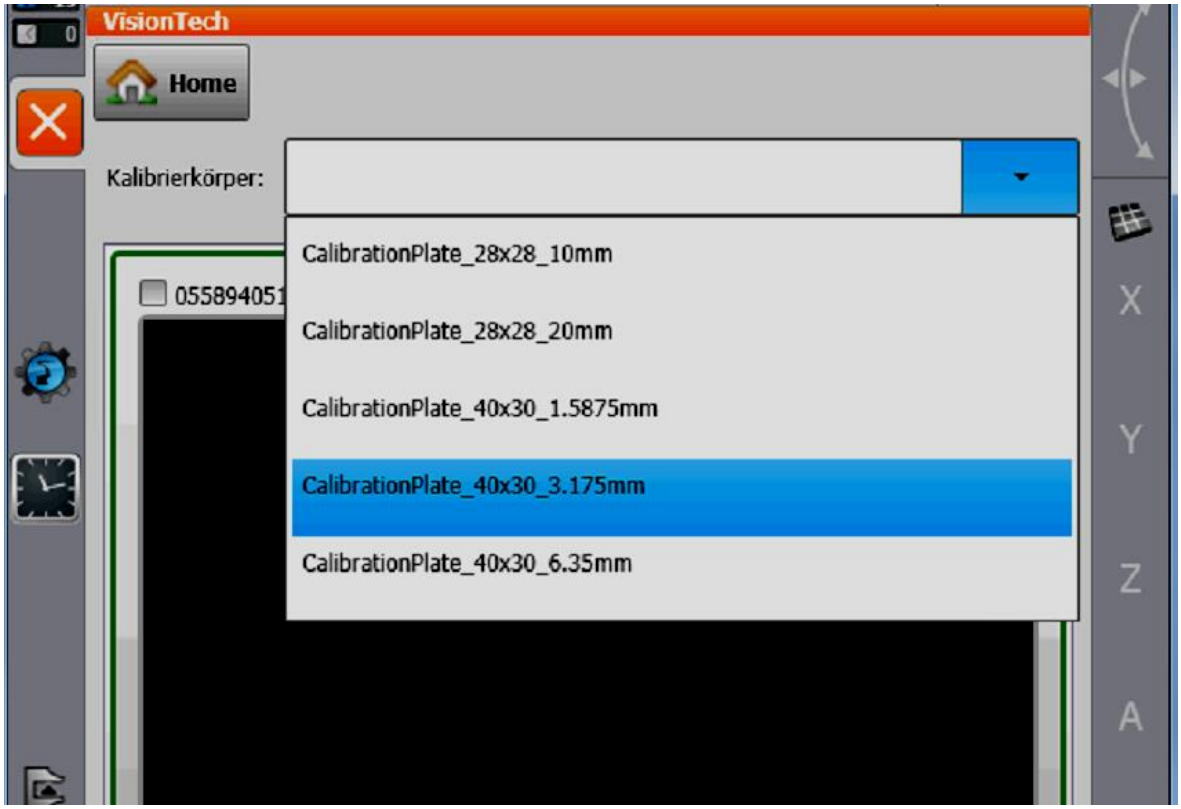

Bild 12: Kalibrierkörper auswählen

Anschließend klickt man den Button **Kalibrierassistent** an und es kann ein Bild aufgenommen werden.

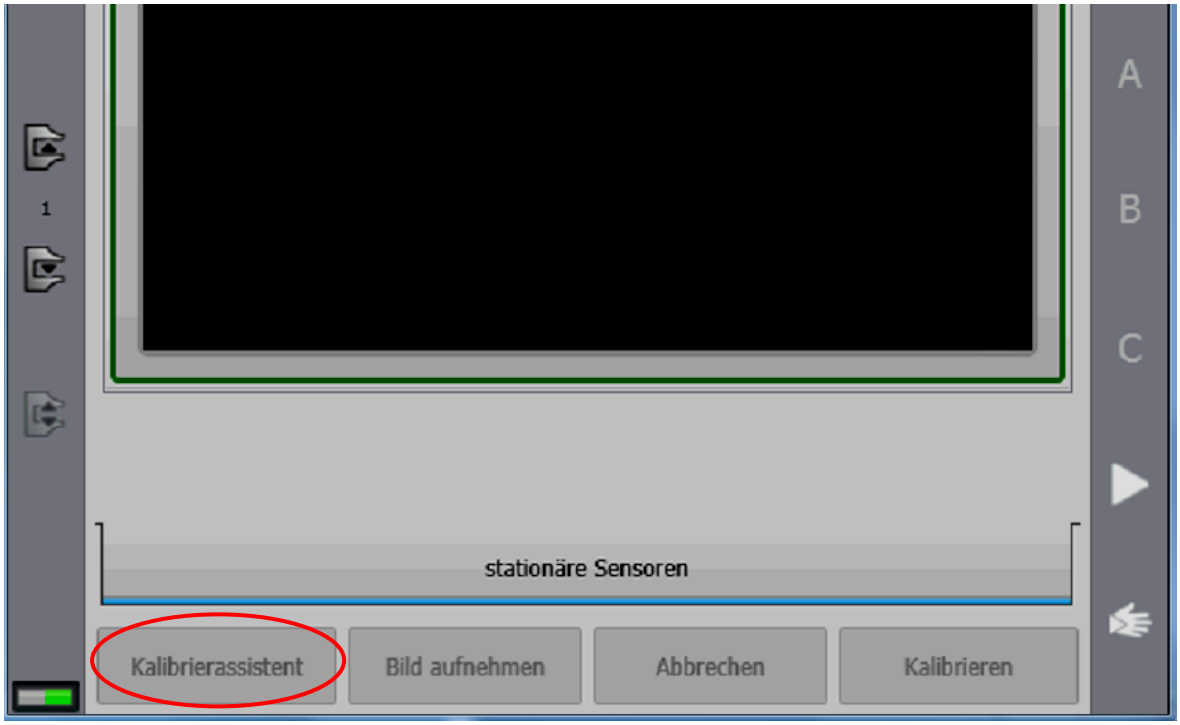

Bild 13: Kalibrierassistent

Durch einen Klick auf den Button **Kalibrieren** öffnet sich ein Fenster in dem man nun einen Namen sowie eine Ebene auswählen kann.

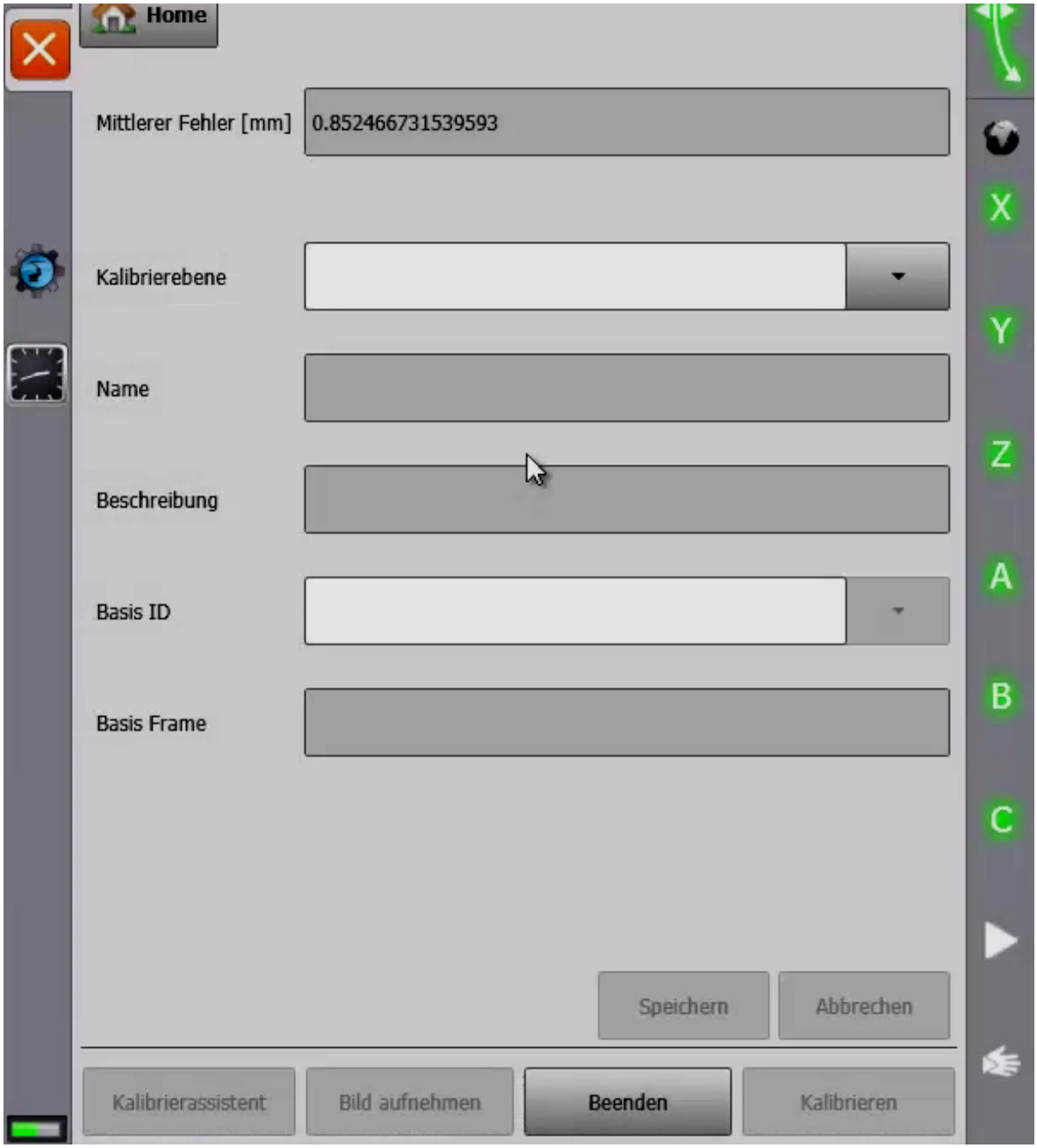

Bild 14: Kalibrierebene anlegen

Die Kalibrierebene kann entweder neu angelegt werden oder eine bereits konfigurierte benutzt werden. Generell ist zu empfehlen, dass bei verschiedenen Objekten auch verschiedene Kalibrierebenen zu benutzen sind, da sonst Fehler im Ablauf der Steuerung auftreten können. Weiterhin ist es empfehlenswert, dass sich die Kalibrierebene nicht auf der Ebene „Tisch“, sondern des zu findenden Musters befindet (Bild 15).

Somit kann die Kamera direkt am Objekt die Koordinaten berechnen und es kommt kein Winkelfehler zustande.

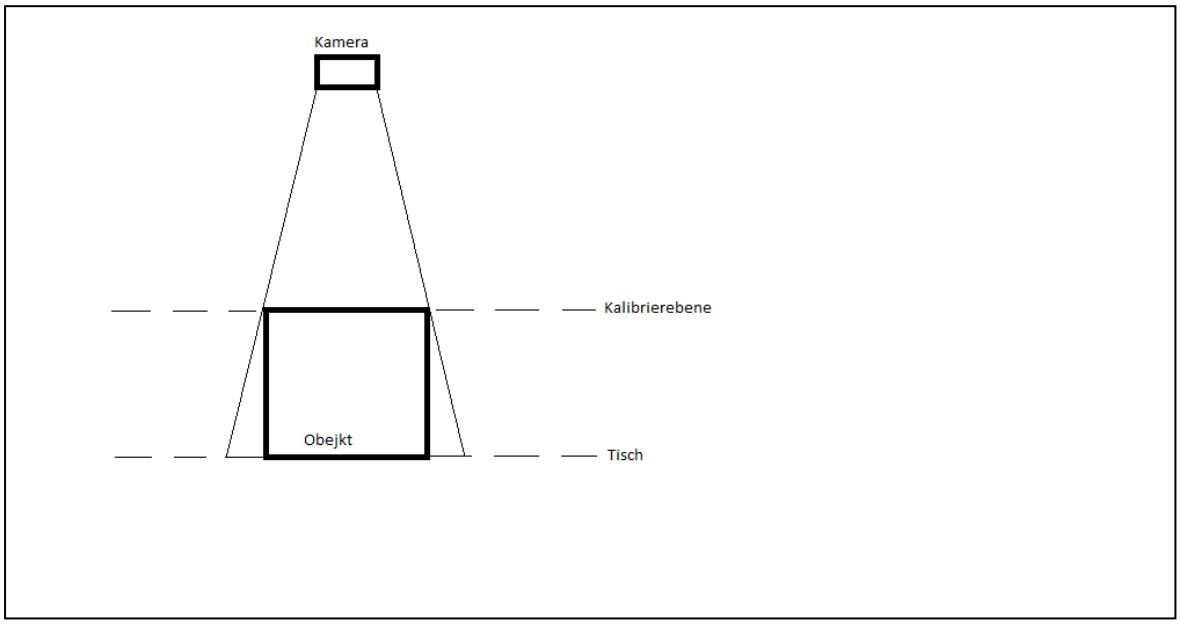

Bild 15: Position der Kalibierebene

Auf dem Bild 15 ist zu sehen, dass es, wenn die Kalibrierebene auf der Ebene des „Tisch“ ist zu Abstandsfehlern kommt. Dann wird vielleicht das Objekt gefunden aber an falscher Position gegriffen. Ist die Kalibrierebene aber an der obersten Kante des Musters angelegt worden, entstehen bei entsprechender Genauigkeit nur zu vernachlässigende Fehler.

Erstellt man also eine neue Ebene, so gibt man dieser einen Namen. Es bietet sich an mit Datum oder mit Prozessständen zu arbeiten, da hier die chronologische Zuordnung einfacher ist. Danach kann man eine Beschreibung hinzufügen und eine Basis auswählen. Diese Basis bezieht sich auf den Roboter. Man muss sich also vorher im Klaren sein in welcher Basis man arbeiten möchte. Auch hier kann man aber auch eine neue Basis_ID erarbeiten. Dieses Einmessen der Base geschieht über die sogenannte 3-Punkt-Methode. Hierzu werden nacheinander erst der Ursprung, dann ein Punkt auf der x-Achse und ein Punkt auf der y-Achse definiert.

Um die Kalibrierung durchzuführen legt man die Kalibrierplatte unter die Kamera und macht ein Foto. In diesem Schritt wird bereits das Koordinatensystem definiert und muss dann auch mit dem Roboter eingehalten werden. Falls es also eine bestimmte Ausrichtung haben sollte, so muss das geschehen bevor das Bild geschossen wird. Diesem Foto wird ein mittlerer Fehler zugerechnet und dient als Referenzbild

für das Projekt. Durch die Erkennung und die bekannte Kästchengröße der Kalibrierplatte wird nun das Bild entzerrt. Die Kalibrierplatte darf nun nicht bewegt werden. Die Ausrichtung des Koordinatensystems erfolgt durch das sogenannte Fiducial. Die lange Seite beschreibt die positive x-Achse und die kurze Seite zeigt die positive y-Richtung an.

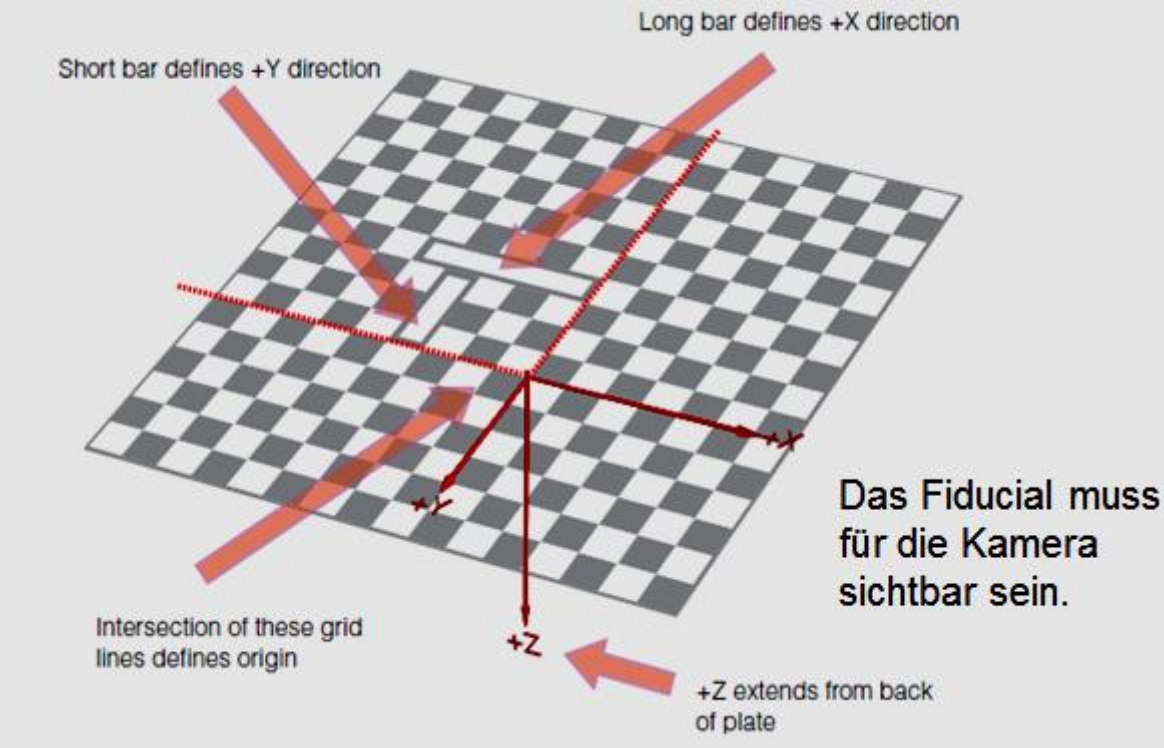

Bild 16: Fiducial [2] Folie 47

Folgt man den Anweisungen des Programms, fährt man den TCP an. Dieser liegt immer an den verlängerten Achsen der X- und Y-Achse. Daraufhin fährt man entlang der X- Achse und danach der Y- Achse. Es ist darauf zu achten, dass man mind. 5cm mit dem Roboter fährt, da sonst die unterschiedlichen Positionen nicht als solche erkannt werden. Ist das geschehen, kehrt man ins VisionTech und die Kalibrierung zurück und speichert diese Kalibrierung.

Die Kalibrierung der Kamera und der Roboterbasis ist damit abgeschlossen. Im Menü im VisionTech kann man das Ergebnis nochmals einsehen. Mit einem Klick auf Kalibrierverwaltung und der Auswahl der Ebene sieht man die Beschreibung und andere Daten. Als nächster Schritt steht nun das Einlernen des Musters an.

3.2 Task- und Toolblockerstellung

- Taskkonfiguration → neuen Task erstellen (siehe Bild 3)
- Referenzbilder machen (an folgendem Speicherort hinterlegt: C:///KRC//TP//VisionTech//Snapshots//Projekt)
- Referenzbilder per Dateiübertragung auf Partner-PC laden
- WorkVisual starten

Tasks sind einzelne Aufgabenbereiche. Sie werden mit speziellen VisionTech-Befehlen im KRL-Programm aufgerufen, wenn es um die Bildverarbeitung geht. Hinter diesen Tasks steckt ein sogenannter Toolblock, der die Bildverarbeitung enthält. Ein Task wird im Visionmenü angelegt und der zugehörige Toolblock wird im WorkVisual vorbereitet. Dies geschieht auf der Windows-Seite der Steuerung und wird später auf die Roboterseite transferiert und im Programm von der Steuerung aufgerufen. Im Toolblock werden Muster eingelernt und Laufzeitparameter eingestellt.

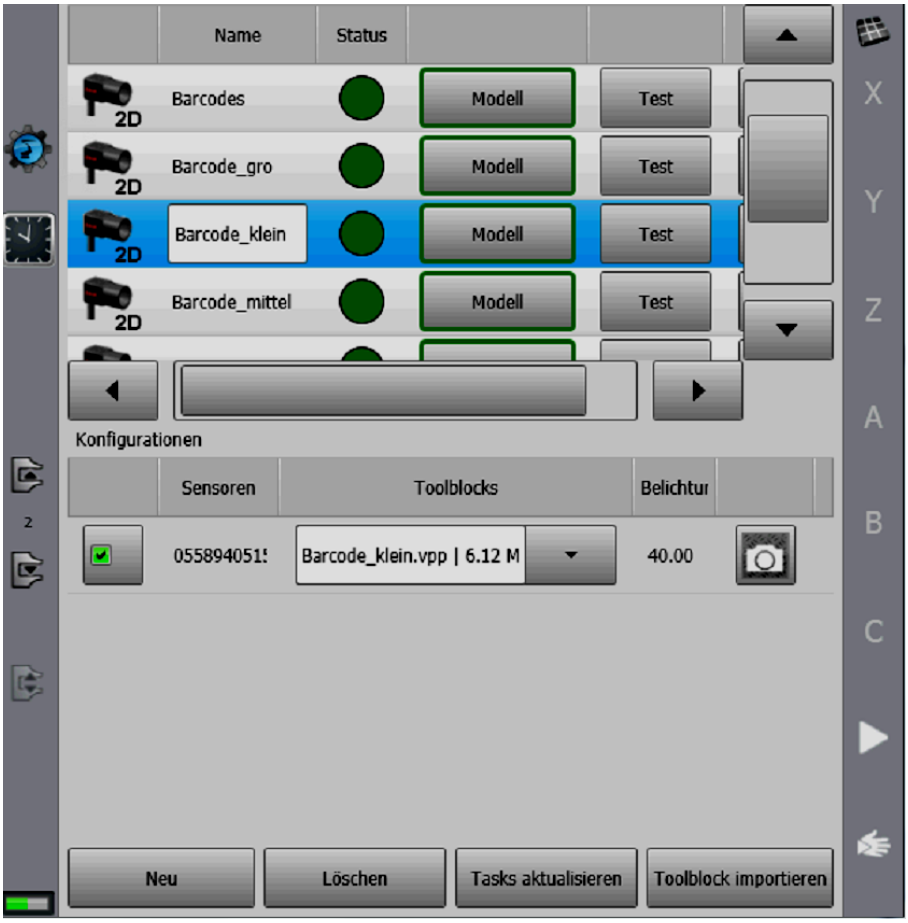

Bild 17: Task anlegen

Mit einem Klick auf das Robotersymbol öffnet sich wieder das Vision-Tech-Menü. Hier kann man auf die verschiedenen Tasks zugreifen, mit einem Klick auf **Taskkonfiguration**.

Es gibt einen Überblick über die bereits vorhandenen Tasks. Klickt man auf einen davon so erscheint der dafür erstellte Toolblock in der Konfiguration. Ist noch kein geeigneter Task vorhanden muss ein neuer erstellt werden. Das geschieht, indem man auf den Button **Neu** klickt und einen Namen vergibt.

Somit ist ein Task erstellt und man kann den Toolblock bearbeiten. Für das Einlernen eines Musters ist es nötig ein Referenzbild zu erzeugen. Für Mustererkennungen empfiehlt es sich mehrere Bilder zu schießen. Im WorkVisual gibt es eine ganze Auswahl von Bildern zum Testen und man kann relativ schnell und einfach auf Störfaktoren reagieren und diese beseitigen. Es können jedoch maximal 20 Bilder in einem Toolblock verwendet werden. Im Bild der Taskkonfiguration gibt es bei der Toolblockauswahl rechts ein kleines Kamerasymbol. Klickt man darauf, bekommt man ein Livebild und man kann die benötigten Musterbilder aufnehmen. Der Speicherungsprozess erfolgt automatisch auf der Roboterseite. In folgendem Pfad werden die sogenannten Snapshots im anzulegenden Ordner gespeichert:

C://KRC//TP//VisionTech//Snapshots//“eigenes Projekt“

Sämtliche Dateien müssen über die **Dateiübertragung** von der einen auf die andere Seite übertragen werden. Mit einem Klick auf den kleinen Pfeil an der Oberseite des Interface gelangt man in dieses Fenster der Dateiübertragung.

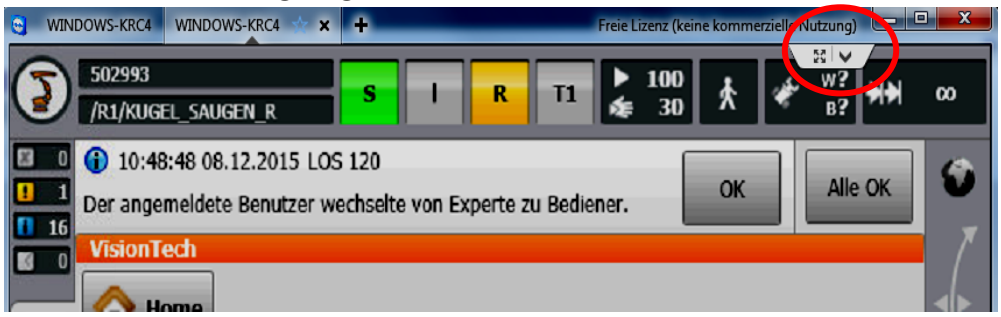

Bild 18: Dateiübertragung

Die Steuerungsseite des Roboters befindet sich auf der rechten Seite und die Bildverarbeitungsseite befindet sich auf der linken. Fotos, die als Referenzbilder gespeichert wurden, müssen also von der rechten

Seite aus dem Snapshotsordner markiert und in den entsprechenden Ordner auf der linken übertragen werden.

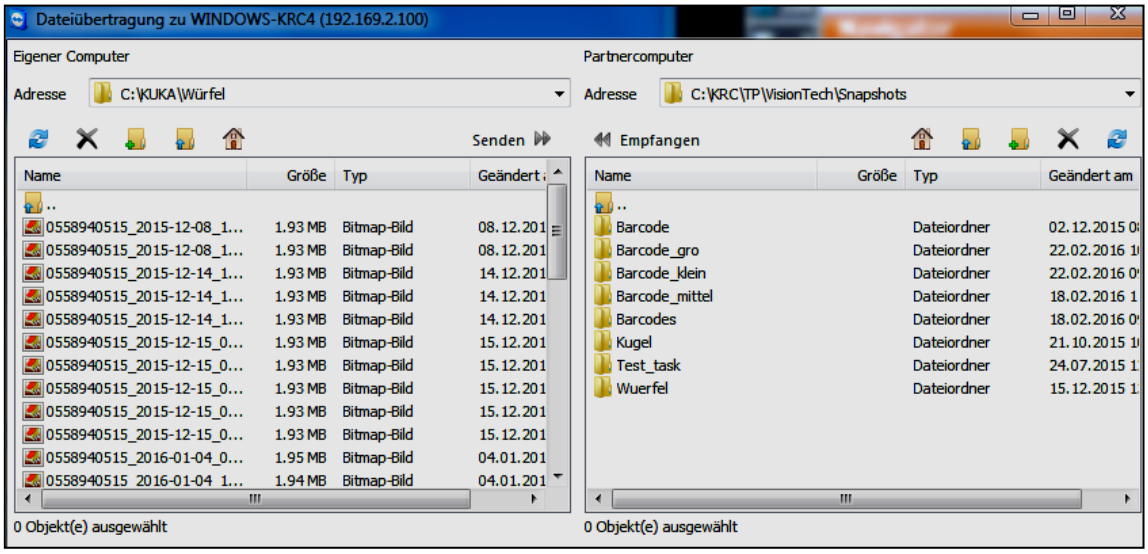

Bild 19: Snapshotordner

Ist das passiert, beginnt man mit der Erstellung der Toolblocks. Das passiert nun im WorkVisual.

3.2.1 Mustererkennung

- **VisionTech öffnen** klicken
- Neuen Toolblock erstellen
- Toolblock→neu→2D
- Bildquellen auswählen
- **Toolbox anzeigen** öffnen
- Kuka → 2D → LocatePartsOneStage per Drag & Drop einfügen
- Verbindungen zwischen Inputs und Outputs herstellen
- **Ausführen** klicken und „LocatePartsOneStage“ und „PatMax“ öffnen
- **Aktuelles Bild als Referenzbild speichern** anklicken und Grafiken bearbeiten
- **CurrentTrainImage** einstellen
- Muster einrahmen/nachzeichnen und Ursprung zentrieren

- Muster trainieren
- Laufzeitparameter einstellen
- „LocatePartsOneStage“ schließen → **gesamtes Werkzeug** speichern und Programm **Ausführen**

Es muss also zuerst ein neuer Toolblock angelegt werden. Hierzu werden mit einem Tastendruck auf den **VisionTech-öffnen**-Button die Werkzeuge und die verfügbaren Inputs und Outputs angezeigt. Danach erstellt man einen neuen Toolblock. Es wird dazu auf die Schaltfläche Toolblock→Neu→2D stationär geklickt.

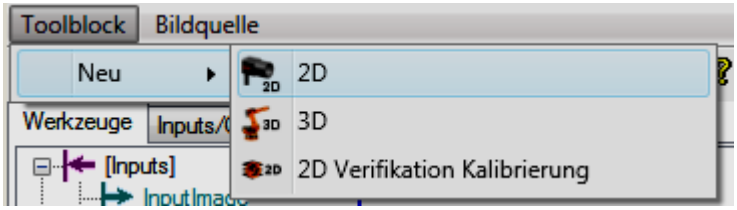

Bild 20: Toolblock anlegen

Anschließend muss eine Bildquelle ausgewählt werden. Mit Hilfe dieser Grafik wird später das Muster eintrainiert. Hier muss man sich entscheiden, ob man ein einzelnes Bild oder einen ganzen Ordner zum Einlernen benötigt. Im Fall, dass ein gesamtes Verzeichnis ausgewählt wird, werden die Ergebnisse theoretisch besser und genauer.

Hat man ein Bild oder ein Verzeichnis gewählt, wählt man den Button **Toolbox anzeigen**

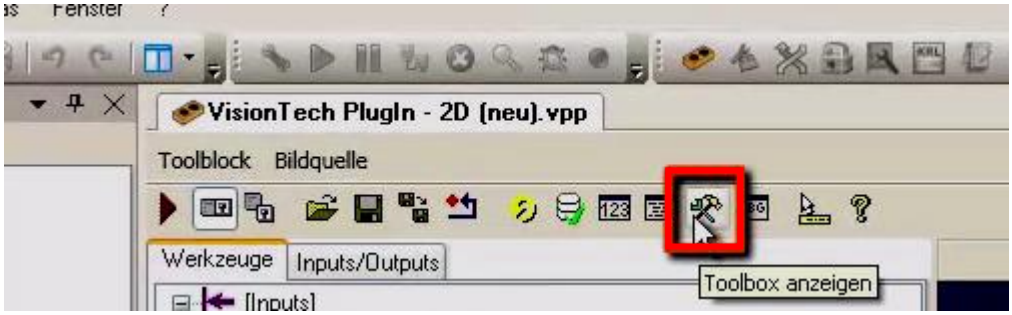

Bild 21:Toolbox öffnen

Danach öffnet sich diese Toolbox und man wählt sich die gewünschte Funktion aus. Hier ist das die Funktion LocatePartsOneStage.

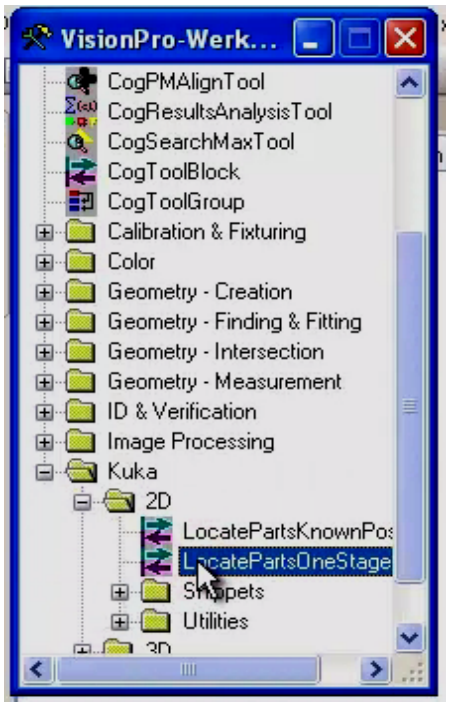

Bild 22: LocatePartsOneStage

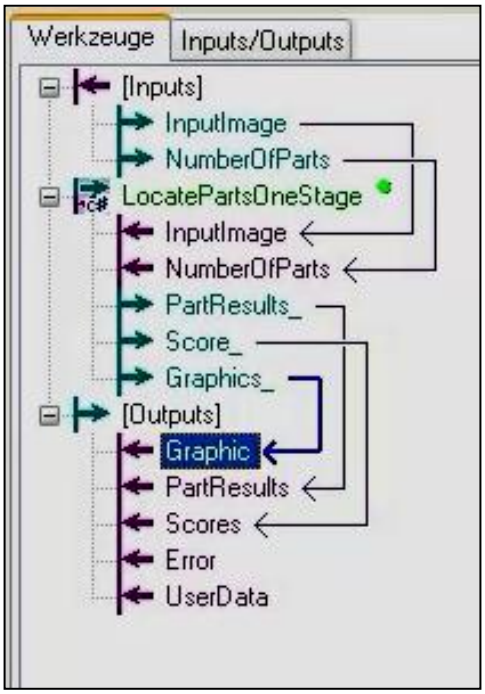

Bild 23: Verknüpfung Inputs→Outputs

Mit der Funktion „LocatePartsOneStage“ ist es möglich Muster in 2D zu erkennen. Diese gilt es nun in den Erkennungsprozess einzubauen. Mit einem Druck der linken Maustaste zieht man sich die Funktion in den gerade angelegten Toolblock und zwar zwischen die Inputs und Outputs (Bild 23). Der Ablauf der Routine wird hiermit festgelegt. Ist es also nötig, dass vorher eine andere Prozedur vollzogen wird, dann müsste man diese vor die Mustererkennung setzen.
Stand jetzt kann man zwar eine Geometrie trainieren, sie würde aber nicht von der Steuerung abgearbeitet und gefunden werden. Hierzu müssen die einzelnen Aufgaben erst miteinander verknüpft werden (Bild 23).

Also stellt man eine Verbindung der einzelnen Bereiche her und bringt so das Programm in einen ablaufbereiten Zustand. Danach klickt man auf die Schaltfläche **Ausführen**  und der Toolblock ist bereit zum Einlernen eines Teils. Es wird nun das gewünschte Bild zum Einlernen angezeigt. Mit mehrmaligen Klicks auf den Button **Ausführen** wird zwischen den verfügbaren Bildern gewechselt und man kann sich das Bild aussuchen, welches am besten für die Erkennung geeignet ist. Im Funktionsbaum kann man sehen, dass noch kein erfolgreiches Training stattgefunden hat (siehe Bild 24).

Ist das Einlernen geschehen, so ist ein grüner Kreis hinter der entsprechenden Funktion (zu diesem Zeitpunkt entsprechend ein roter).

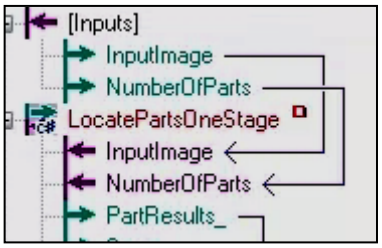

Bild 24: Muster nicht trainiert

Zum Einlernen eines Musters klickt man doppelt auf „LocatePartsOneStage“ und es öffnet sich ein Fenster.

In diesem gibt es die Funktion „PatMax“ und auch dies öffnet man mit einem Doppelklick. Im sich öffnenden Fenster gibt es nun verschiedene Einstellmöglichkeiten.

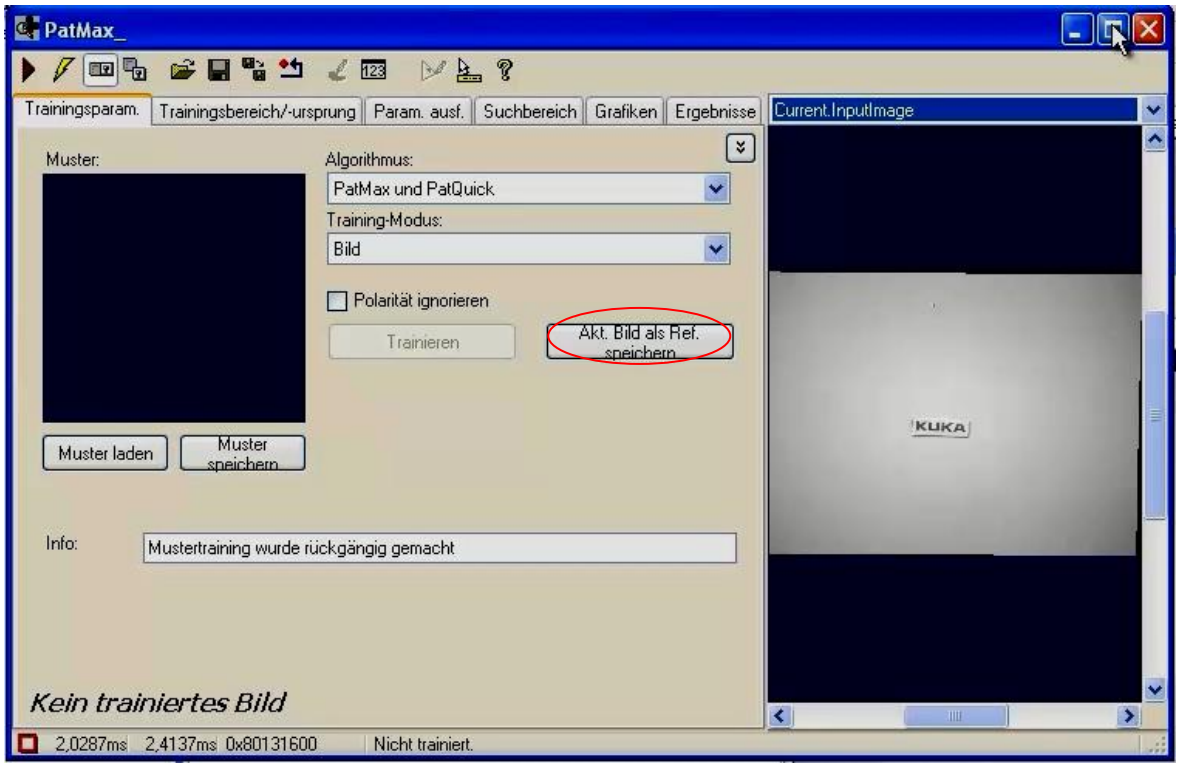

Bild 25: PatMax

Zur besseren Ansicht kann man das Fenster maximieren und mit einem Klick auf die rechte Maustaste auf den Bildbereich das Bild an die jeweilige Fenstergröße anpassen. Das angezeigt Bild muss nun als Referenzbild gespeichert werden (Bild 25). Dieses dient also als Vergleichsbild für andere und muss somit entsprechend gut eintrainiert werden.

Weiterhin kann man vorab einstellen wonach explizit gesucht und welche Ergebnisse angezeigt werden sollen. Das passiert im Reiter „Grafiken“.

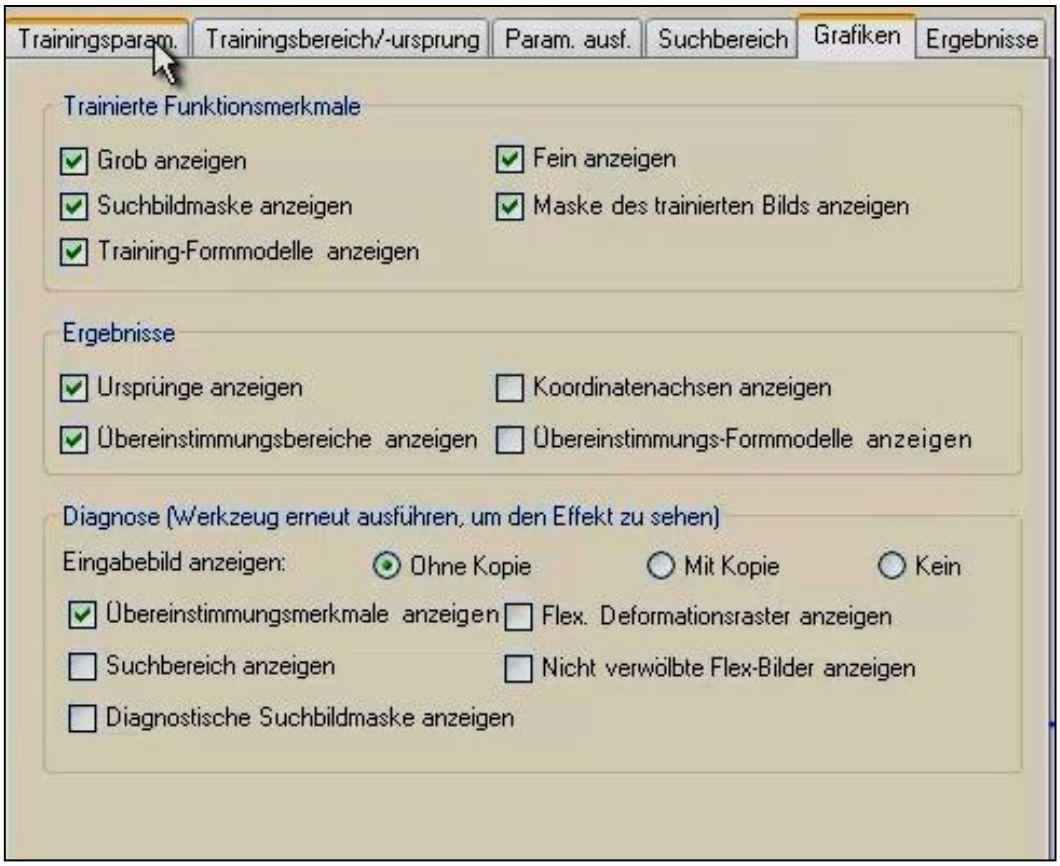

Bild 26: Ergebnisgrafiken einstellen

Im Projekt wurden die Option „Fein anzeigen“, „Grob anzeigen“ und „Übereinstimmungsmerkmale“ angehakt. Den Unterschied zwischen grober und feiner Anzeige kann man erkennen wenn man die Funktion später aus- und einschaltet.

Nun wird das Bild trainiert. Hierzu wird eine Anzeigeauswahl getroffen. Klickt man oben rechts im Fenster auf die aktuellen Bilder kann man zwischen dem aktuellen Input-Image, dem Trainingsbild und dem zuletzt gefundenen wählen. Das aktuelle Bild kann man ändern, indem man das Fenster schließt und das Programm ausführt. Dann werden alle eingeladenen Bilder durchlaufen und man kann sich ein neues Bild aussuchen. Das zuletzt gefundene dient dazu, Ergebnisse nochmal sichtbar zu machen und Funktionen zu testen. Es wird also das aktuelle Trainingsbild ausgewählt.

Bild 27: CurrentTrainImage

Somit wird ein Koordinatensystem angezeigt und eine zuvor ausgewählte Kontur. Man könnte also auch einen Kreis, eine Ellipse oder ein Polygon anzeigen lassen. Da die Körper im Labor eine ungleichmäßige Kontur haben wurde das Polygon ausgewählt. Die Form kann man im Reiter „Trainingsbereich/-ursprung“ ändern. Unter der Funktion „Bereichsform“ kann nun zwischen verschiedenen Formen eine angepasst werden. Mit dem Polygon können auch Punkte hinzugefügt und entfernt werden. Das angezeigte Muster muss nun sehr genau um die zu suchende Kontur gezogen werden. Und zwar so genau, dass die Kontur umschlossen, aber auch nicht zu weit von den Kanten entfernt ist, da sonst Störfaktoren mit einfließen, die nicht berücksichtigt werden sollten. Das Koordinatensystem muss nun zentriert werden. Der Mittelpunkt liegt immer in der Mitte der Kontur. Das bedeutet demzufolge nicht, dass immer der Mittelpunkt des Körpers mit der Kontur übereinstimmt. Die Zentrierung erfolgt im Reiter „Trainingsbereich/-ursprung“ auf dem Button „Ursprung zentrieren“. Im Reiter „Trainingsparameter“ klickt man nun auf **Trainieren** und das Objekt ist eingelernt.

Das trainierte Teil wird nun in den Ergebnissen ausgegeben. Dort werden neben den Längskoordinaten auch die Güte und der Winkel angezeigt. Diese Laufzeitparameter können angepasst werden, falls es nötig ist nur spezielle Teile zu erkennen. Wird eine Güte von 0,5 definiert, so ist das die Mindestanforderung, mit der das gefundene Teil mit dem eingelernten übereinstimmen muss. Liegt die Güte im Prozess unter 50%, so wird es nicht in den Ergebnissen angezeigt und das Programm wird beendet. Durch Verzerrungen ist es möglich, dass das Objekt größer oder kleiner erscheint als es tatsächlich ist. Diese Einstellung kann man über den Maßstab regulieren, was jedoch nur selten nötig ist. Im Reiter „Parameter ausführen“ können diese Daten eingestellt werden.

Bild 28: Parameter ausführen

Der ausgegebene Winkel hat Auswirkungen auf die Anfahrt des Roboters, nachdem das Objekt erkannt wurde. Es ist also nötig, dass man eine Anfahrt mit dem Roboter simuliert. Dazu wird das Objekt in den Suchbereich gelegt und man fährt mit dem Roboter so an den Bildbereich heran, wie es später im Programm passieren soll. Nun lässt man sich die aktuellen Koordinaten des Roboters anzeigen. Im VisionTech-Menü klickt man dazu auf **Anzeige**. Dort kann zwischen aktuellen und kartesischen Anzeigen wählen. In der Winkelanzeige A sieht man nun die Orientierung. Im Projekt lag dieser Winkel bei ca. 0 Grad. Ist das Objekt z.B. ein Würfel, so kann die Erkennung nur einen Unterschied von 90 Grad erkennen. Es ist also nicht nötig eine Anzeige von vollen 360 Grad auszugeben, da sich die Würfelform alle 90 ° wiederholt. Von diesem Beispiel ausgehend liegt es dementsprechend nahe, dass die Winkelangabe zwischen -45 und +45 Grad liegen sollte. Bei einer Kugel wiederum ist eine Winkelangabe nicht notwendig, da keine Orientierungen anhand von Kanten bestimmt werden können. Schaltet man die Winkelberechnung aus (mit einem Klick auf den Pfeil neben der Anzeige) reduziert sich auch die Prozesszeit der Bildbearbeitung. Diese kann man übrigens die ganze Zeit am unteren Bildschirmrand im WorkVisual überprüfen.

Jetzt ist das Einlernen der Kontur abgeschlossen. Nun werden die Funktionen des „PatMax“ und des „LocatePartOneStage“ geschlossen und das Programm kann ausgeführt werden. Dort wird das gefundene Muster mit grünen Kanten angezeigt. Hat man mehrere Bilder zum Einlernen genutzt kann man die Ergebnisse von allen einsehen. Es könnte dabei vorkommen, dass es Probleme bei der Erkennung gibt. Entweder wird kein Teil erkannt oder es wird verdreht angezeigt. Ist ein solches Bild vorhanden, kann man dieses nutzen und ändert Parameter, die die Erkennung beeinflussen. Folgende Punkte können interessant dafür sein:

- Muster nachzeichnen
- Güte herabsetzen
- Kontrastschwelle ändern
- Kanten des Objektes kenntlicher machen
- Belichtungszeit ändern
- Maskeneditor bearbeiten

Die Option „Maskeneditor“  im Reiter „Trainingsparameter“ dient dazu bestimme Muster auf dem Objekt herauszufiltern. Löcher, Strukturen oder Muster darauf werden somit nicht mit in die Bildverarbeitung einbezogen. Damit liegt der volle Schwerpunkt auf den äußeren Kanten.

Das Suchen nach den richtigen Einstellungen kann teilweise etwas dauern. Funktioniert das Programm aber einmal, gibt es viele Möglichkeiten die Erkennung durchzuführen. Anschließend wird das gesamte Werkzeug gespeichert.

Trotz der oben genannten Gründe und Problemlösungen, falls die Teile nicht gefunden werden, gibt es noch eine andere Möglichkeit warum Objekte nicht oder schlecht erkannt werden. Das Thema der Beleuchtung wurde bereits in früheren Praktikumsarbeiten angesprochen. Im Labor gibt es neben dem eingebauten Blitzlicht der Kamera zwei künstliche Lichtquellen, den Lichtkranz und die Deckenleuchte des Raumes. Tabellen im Anhang sollen die Ergebnisse der Beleuchtungstests darstellen. Sind die Deckenleuchte und der Lichtkranz an und herrscht eine starke Sonneneinstrahlung kann es durchaus zu Problemen bei der Mustererkennung kommen, es ist aber eher nicht zu erwarten. Genauso sieht es aus wenn die Leuchten an sind und der Himmel bewölkt ist. Einzelne Ausfälle kommen vor, sind aber selten. Anders sieht es aus, wenn starke Sonneneinstrahlung vorherrscht und die Lampen aus sind. Dadurch, dass nicht die Lichtquellen von oben die Mustererkennung unterstützen, sondern die komplette Einstrahlung seitlich eintrifft, kommt es zur Schattenbildung im Suchbereich. Somit kann manchmal selbst die sehr gute Kamera nicht den Unterschied zwischen Kante und Schatten erkennen. Für fünf erkannte Teile wurden sieben Versuche benötigt und ein Ergebnis ist zwar akzeptabel mit knapp 78 % erkannt worden, kann aber auch zu einer falschen Position führen. Bei bewölktem Wetter und ausgeschaltetem Licht waren die Ergebnisse besser, jedoch war ein Ergebnis unbrauchbar und ein Teil wurde nicht gefunden. Die Versuche zeigen, dass es nicht schadet eine oder mehrere zusätzliche externe Lichtquellen zu installieren, um somit den Suchbereich komplett auszuleuchten.

3.2.2 Barcodeerkennung

In diesem Abschnitt wird erläutert, wie ein Barcode auf einem bestimmten Objekt oder in einem Bildbereich gefunden und ausgegeben wird. Hierzu werden zuerst die beiden Hauptfunktionen analog zur Funktion „LocatePartsOneStage“ in das Ablaufprogramm gezogen.

- „LoopPresenceCheck“ per Drag & Drop einfügen
- „StringsFromLists“ per Drag & Drop einfügen
- Verbindungen zwischen Inputs und Outputs herstellen
- „LoopPresenceCheck“ öffnen und Verbindungen zwischen Inputs und Outputs herstellen
- „ProcessFixturePart“ öffnen und „CogBarcodeTool“ einfügen
- Verbindungen erstellen und „CogBarcodeTool_1“ öffnen
- Codetyp einstellen
- Bereichsform und Suchbereich einstellen
- Terminals hinzufügen (Results →Item(0) → OwnedDecoded → CogBarcodeResultDecoded → String <String>) und **Ausgabe hinzufügen**
- „ListCreator“ und „GraphicCollector“ einfügen
- „BarcodeString“ der Liste hinzufügen
- „LoopPresenceCheck“ schließen und Einstellungen testen
- Toolblock übertragen und dem eigenen Task zuweisen

Um die Routine auszuführen wird eine Funktion benötigt. Hierzu wird wieder die Toolbox  geöffnet. Im Strukturbaum findet man unter KUKA→2D→Snippets→1_SampleLoops die Funktion „LoopPresenceCheck“. Wie bereits mit der vorherigen Funktion zieht man diese nun zwischen „LocatePartOneStage“ und den Outputs. „LoopPresenceCheck“ dient der Erkennung eines Barcodes. In diesem werden Suchregionen, Arten des Codes und auszugebende Elemente als Strings eingestellt.

Um den gefundenen Barcode auszugeben ist es nötig den zuvor erstellten String aus einer Liste herauszulesen und an die Robotersteuerung zu senden. Das Herauslesen erfolgt mit Hilfe der Funktion „StringFromLists“.

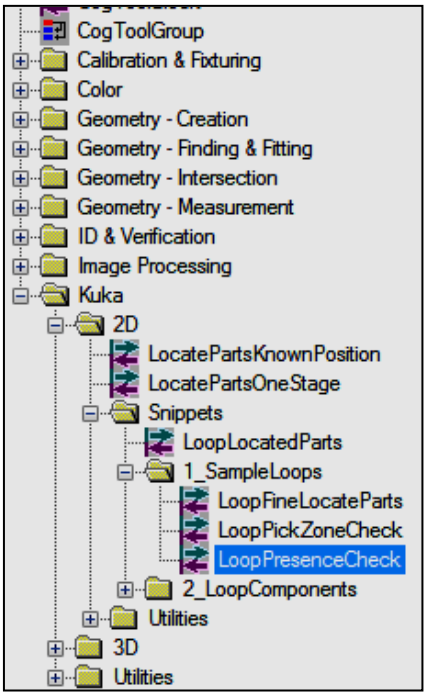

Bild 29: LoopPresenceCheck

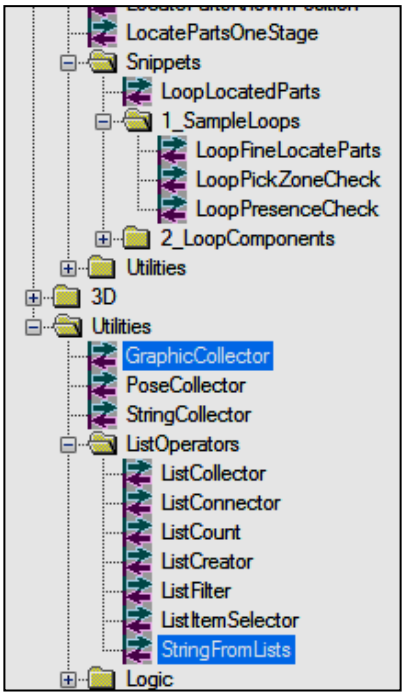

Bild 30: StringsFromLists

Damit die Funktionen nacheinander abgearbeitet werden, müssen diese wieder miteinander verknüpft werden.

Kurz erläutert bedeutet diese Ansicht (Bild 31), dass ein Ergebnis samt Koordinaten und Güte aus der Mustererkennung an „LoopPresenceCheck“ übergeben wird.
Dort wird der ermittelte Barcode als String in eine Liste geschrieben. Um den String wieder aus der Liste herauszubekommen wird die Liste an die Funktion „StringFromLists“ übermittelt und von dort wird der String an die UserData gegeben, welche im KRL-Programm angezeigt wird. Dazu wird im Punkt 3.3 Bezug genommen.

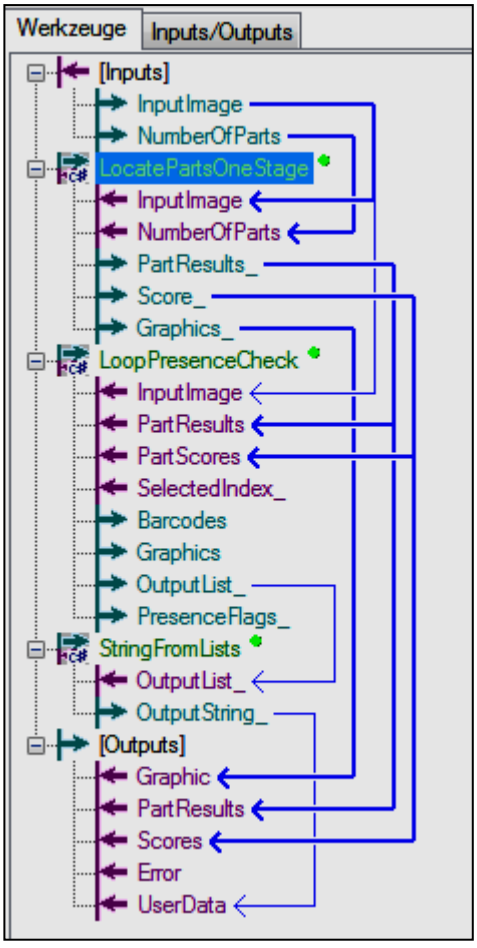

Bild 31: Strukturbaum

Öffnet man den „LoopPresenceCheck“, dann kommt ein weiterer Strukturbaum zum Vorschein. Hier sind ebenfalls Verknüpfungen notwendig, wie das folgende Bild 32 veranschaulicht.

In diesem Fenster werden viele Einstellungen getroffen, die für das Suchen, Erkennen und Ausgeben des gefundenen Codes verantwortlich sind. In der Funktion „ProcessFixturePart“ sind Einstellungen für das Erkennen eines Barcodes enthalten. Dort wird auch der Barcode in einen String umgewandelt. Dieser String wird in der darauffolgenden Funktion „ListCreator“ in eine Liste geschrieben und von dort direkt der Ausgabe hinzugefügt.

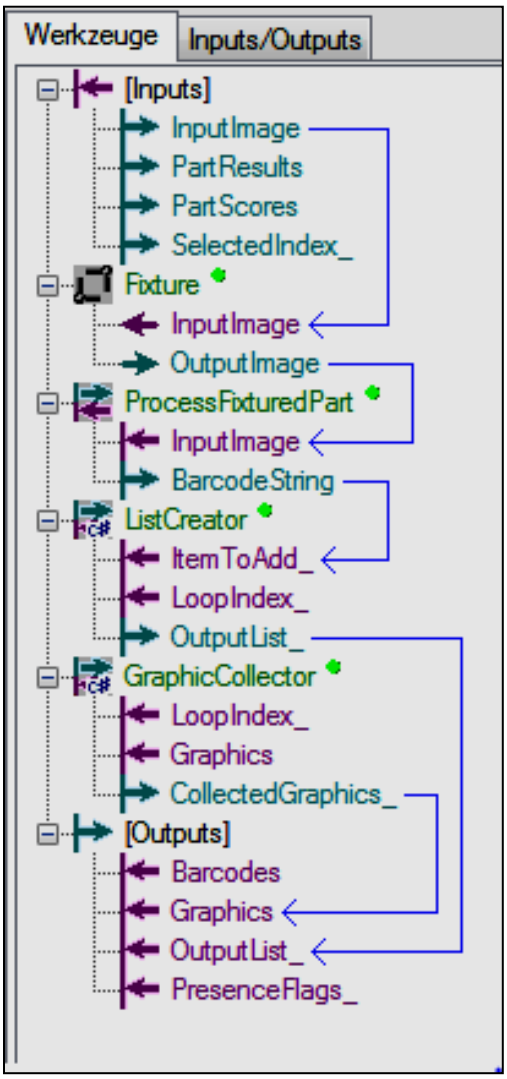

Bild 32: ProcessFixturePart

Mit einem Doppelklick auf „ProcessFixturePart“ öffnet sich das Einstellfenster für die Barcodeerkennung. Dort muss zuerst noch ein Tool eingefügt werden. Aus der Toolbox holt man sich aus dem Ordner „ID&Verification“ das „CogBarcodeTool“ und fügt es zwischen den Inputs und den Outputs ein. Danach wird das Tool geöffnet.

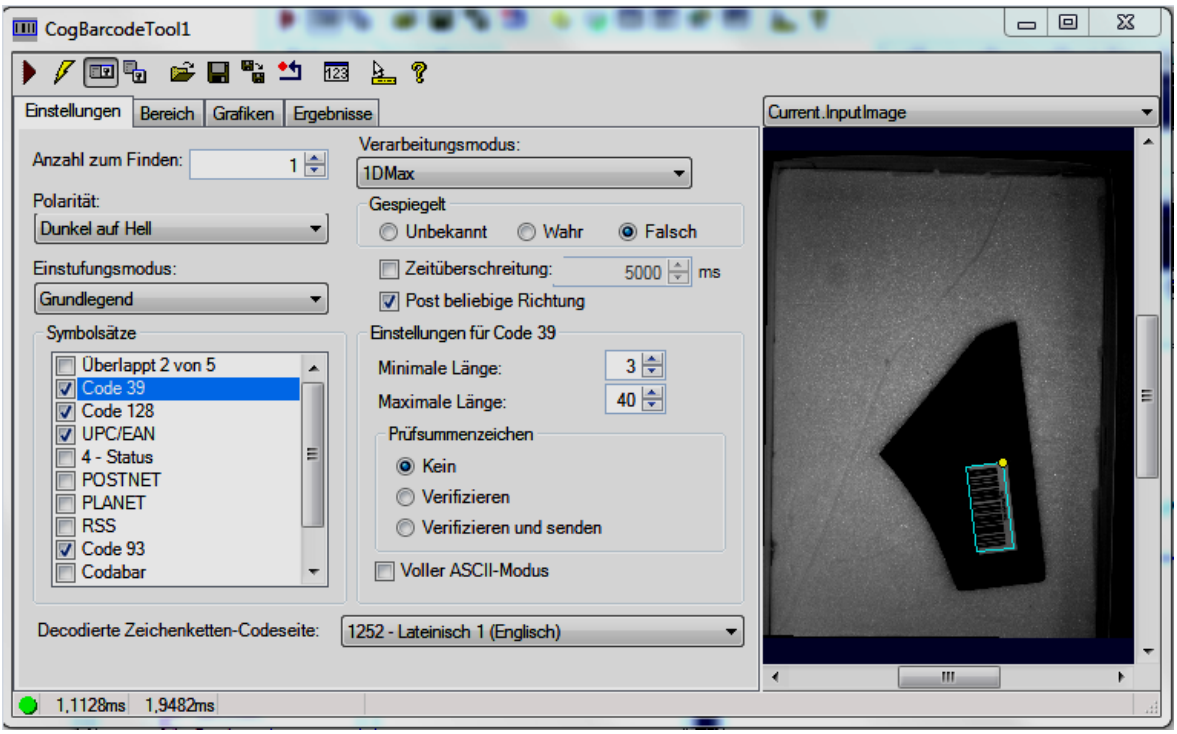

Bild 33: CogBarcodeTool

Im Reiter „Einstellungen“ wird zuerst die Art des Barcodes eingestellt. Kommen verschiedene Arten zum Einsatz hakt man diese an. Im Projekt wurde der „UPC/EAN“ verwendet. Damit wäre es ausreichend nur diesen auszuwählen.

Anschließend wird der Bereich, in welchem gesucht werden soll, eingegrenzt. Hier ist es wieder möglich eine Bereichsform auszuwählen. Mit der Form „CogPolygon“ können Punkte hinzugefügt oder gelöscht werden. Es ist darauf zu achten, den Barcode sehr genau zu markieren. Ist der Barcode auf ein Papier gedruckt und dieses aufgeklebt kann es passieren, dass die Papierkante als weitere vermeintliche Nummer erkannt wird. Das hat zur Folge, dass der Code nicht erkannt wird, da er eine nicht gültige Anzahl an Zeichen hat.

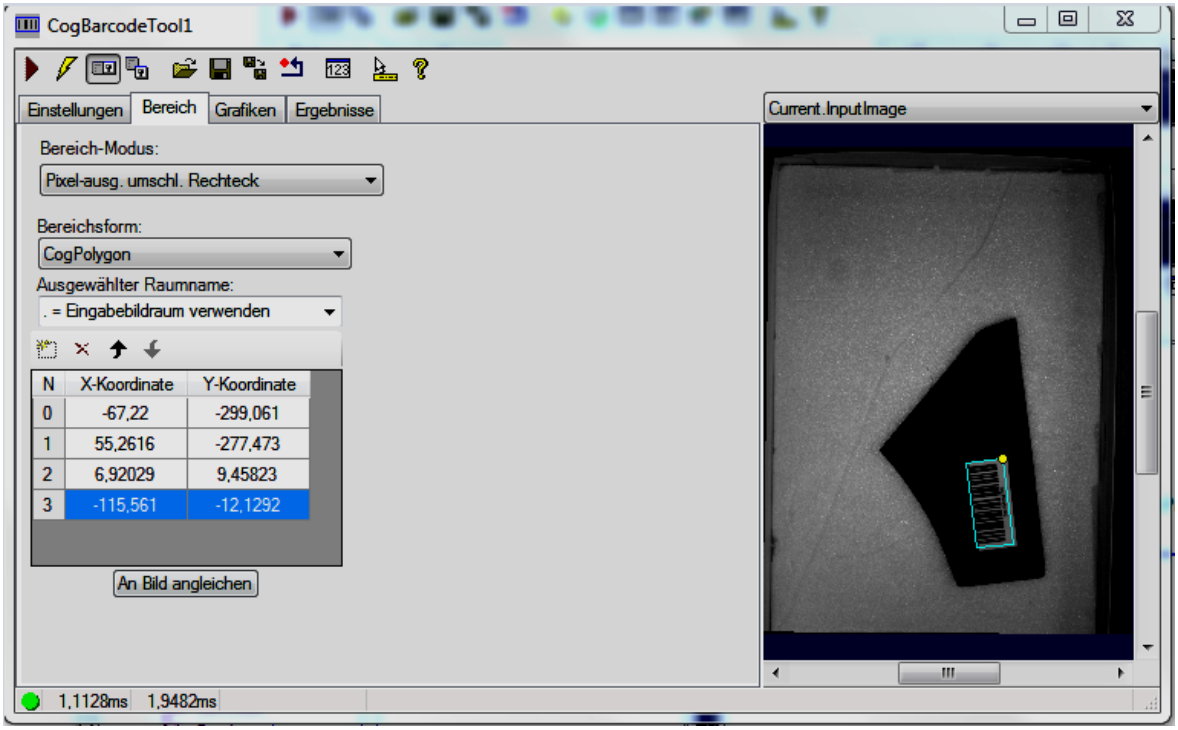

Bild 34: Bereich des Barcodes

Sind alle Einstellungen getroffen kann das Tool geschlossen werden und man beginnt damit, den Code als String zu schreiben und diesen der „Outputlist“ anzuhängen. Hierfür ist es nötig die Resultate mit den Outputs zu verbinden. Im Fenster davor („ProcessFixturePart“) können sogenannte Terminals hinzugefügt werden. Diese Terminals beinhalten unter anderem eine Einstellung den Barcode als String zu definieren. Durch einen Klick mit der rechten Maustaste auf „CogBarcodeTool1“ öffnet man das Fenster „Terminals hinzufügen“.

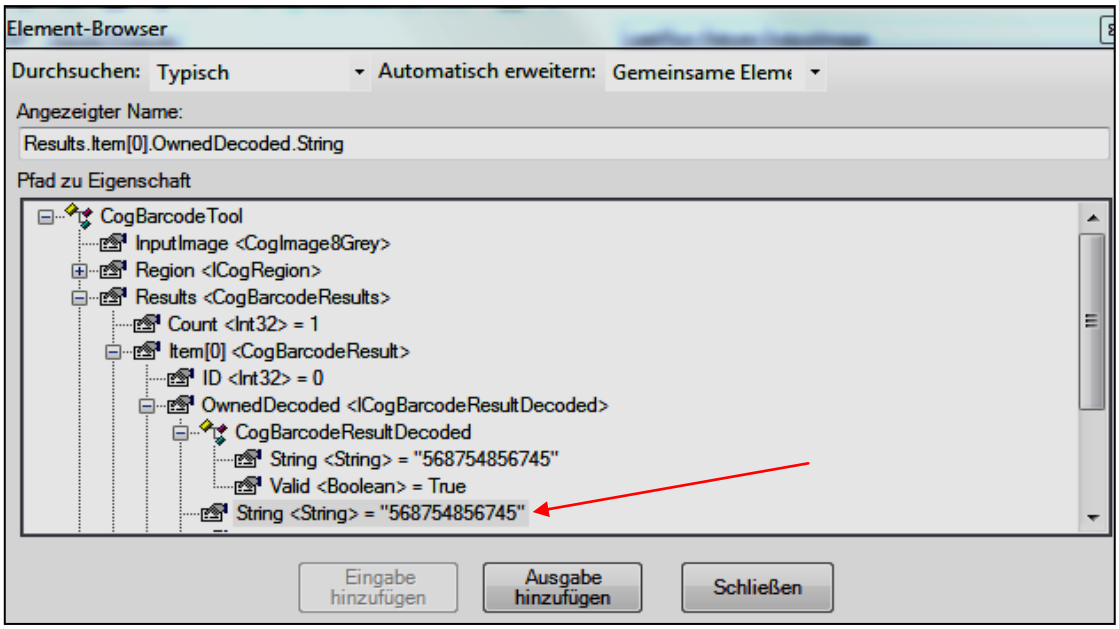

Bild 35: Terminal hinzufügen

Mit einem Klick auf **Ausgabe hinzufügen** wird der dekodierte String den auszugebenden Elementen hinzugefügt. Wie bereits erwähnt wird der String in eine Liste geschrieben. Diese muss aber zuerst kreiert werden. Dazu wird im Fenster „LoopPresenceCheck“ mit der Toolbox der „ListCreator“ eingefügt. In diesem wird über den Input ein Merkmal hinzugefügt und über die Outputs direkt an den Ausgang der Schleife gelegt. Danach wird noch die Funktion „GraphicCollector“ eingefügt und die Grafiken ebenfalls den Outputs angehängt.

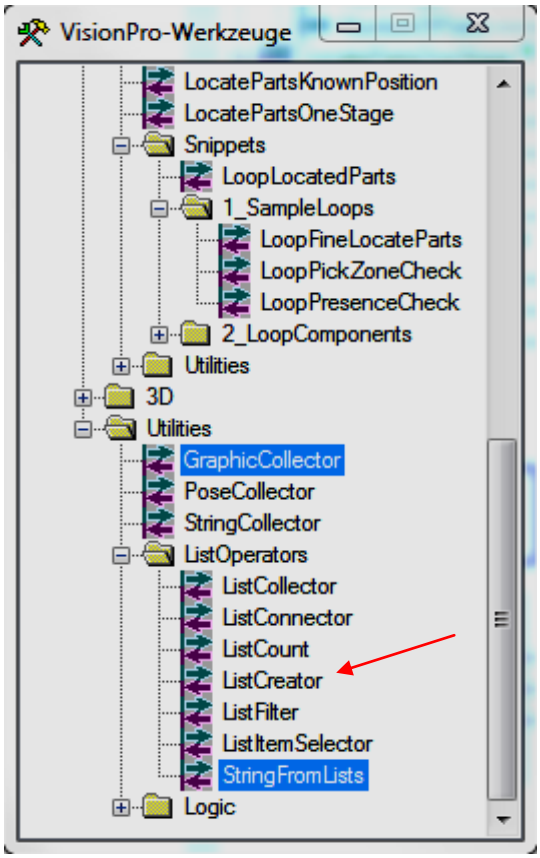

Bild 36: ListCreator

Bis auf die Herstellung der Verknüpfungen sind keine weiteren Einstellungen notwendig. Es können also alle Einstellungsfenster geschlossen werden. Wie im Bild 36 zu sehen ist, ist es nun nötig, die Strings aus der Liste wieder herauszulesen. Dieses Tool wird zwischen dem „LoopPresenceCheck“ und den Outputs der gesamten Bildverarbeitung gezogen. In der Struktur ist zu sehen, dass die ausgegebene Liste (OutputList) als Eingang für die Funktion „StringsFromLists“ genutzt wird. Der ausgelesene String dient als Output und gleichzeitig als Inhalt der UserData. Einstellmöglichkeiten gibt es auch hier nicht. Somit kann man den Toolblock speichern und die Einstellungen testen.

Zum Testen der Einstellungen kann man die einzelnen Funktionen überprüfen. Dazu wählt man zwischen der „PatMax“- und der „LoopPresenceCheck“-Funktion aus. Klickt man den Button **Ausführen** kann man die Ergebnisse jedes Teils, welches zuvor eingeladen wurde, einsehen. Kommt es dabei zu Problemen könnten folgende Punkte hilfreich sein, Lösungen zu finden:

- Suchbereich ausdehnen oder eingrenzen
- Arten des Barcodes eingrenzen
- Barcode neu generieren und großzügiger ausschneiden (Erkennung einer Kante als Strichcode ausschließen)
- Belichtungszeit ändern
- Polarität verändern

Wenn alle Barcodes erkannt werden ist die Erkennung abgeschlossen und der Toolblock kann in die Steuerung geladen werden.

Das passiert über die Funktion der Dateiübertragung. Man holt sich dazu den gespeicherten Toolblock und legt diesen in folgendem Ordner auf der Steuerung ab: C:\KRC\TP\VisionTech\Toolblocks

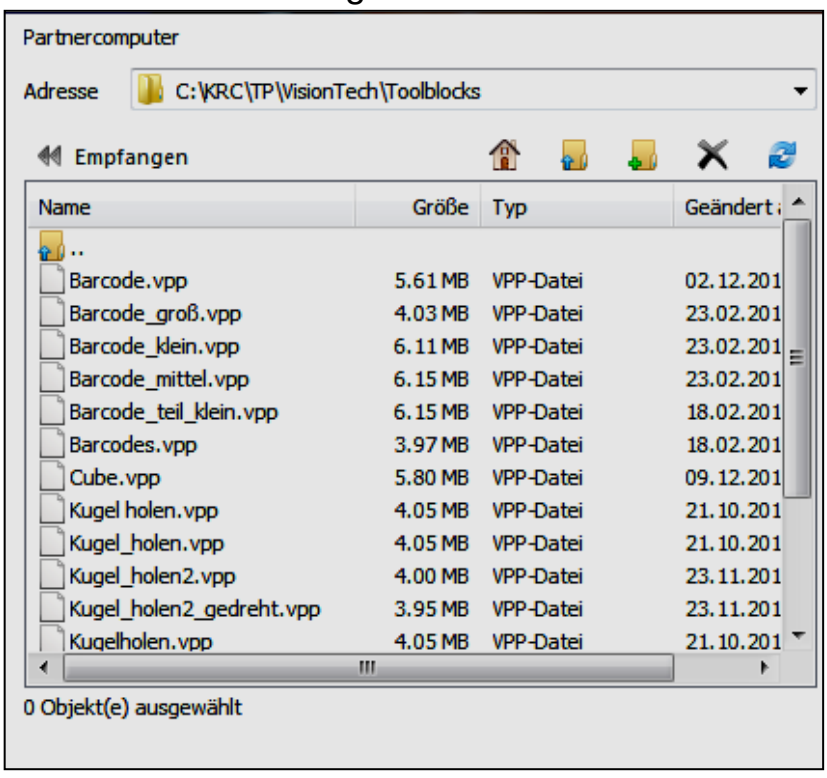

Bild 37: Toolblockordner

Anschließend geht man auf der Steuerung in das VisionTech-Menü auf die Taskkonfiguration. Dort ist es nötig den Toolblock mit dem angelegten Task zu verknüpfen. Dazu wählt man den entsprechenden Task an und klickt dann auf **Toolblock importieren**.

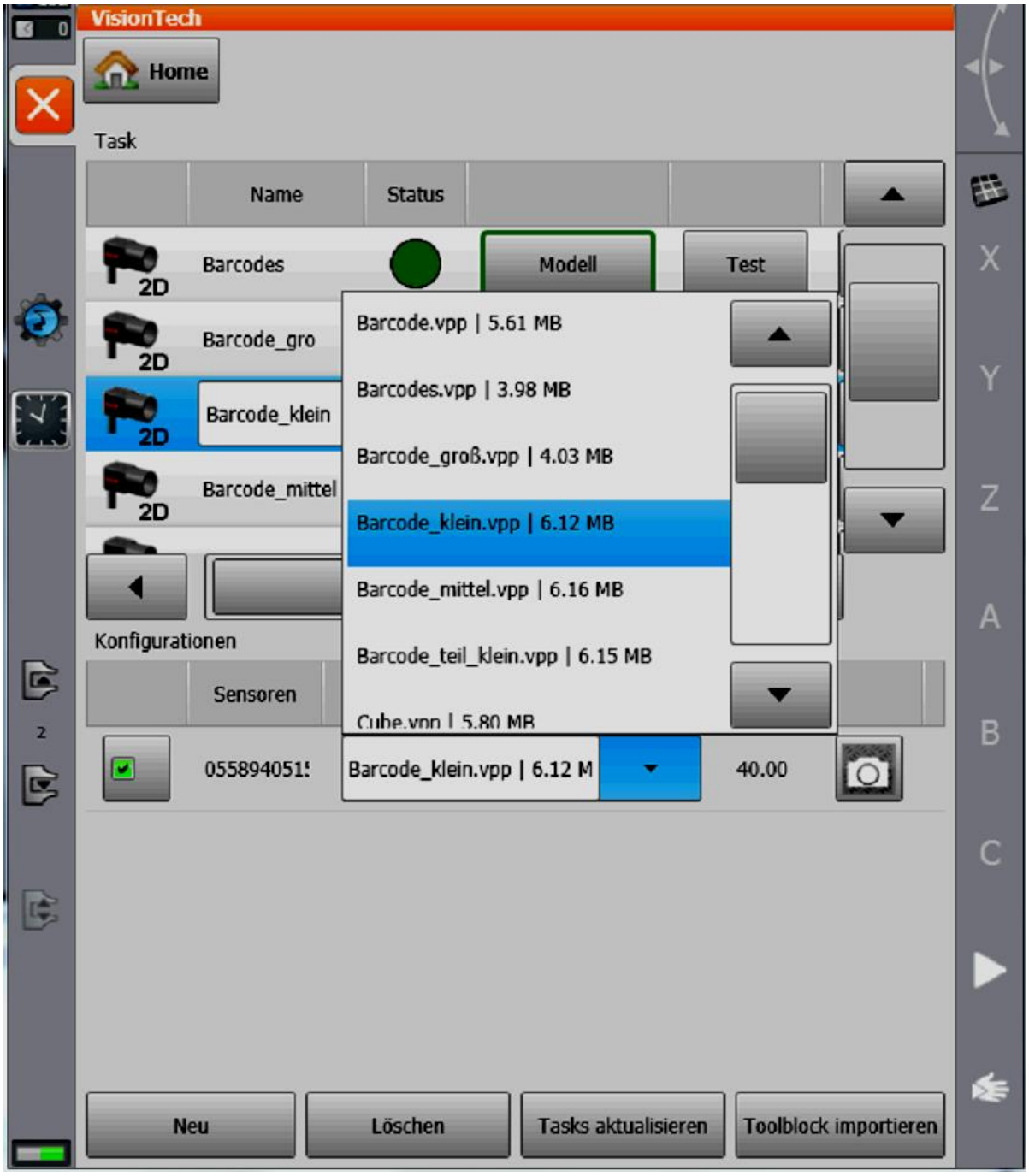

Bild 38: Task auswählen

Ist dieser importiert, dann schadet es nicht den Task zu aktualisieren. Falls später Änderungen am Toolblock vorgenommen werden, so ist es nach der Übertragung nicht mehr nötig den Toolblock zu importieren, sondern nur den Task zu aktualisieren.

Nun ist es nötig ein Modell zu erzeugen. Dafür muss zuerst die Basis entsprechend der Kalibrierplatte eingestellt werden. Dann wird ein Bild gemacht, welches als Referenzmodell dient. Dazu muss das Teil in der Referenzposition liegen. Folgt man den Anweisungen des Fensters und das Foto wird geschossen, so sieht man, dass das Teil erkannt wurde und Koordinaten angezeigt werden. Danach kann die Modellerzeugung geschlossen werden und es kann ein Testlauf gestartet werden. Es ist zu sehen, dass sich das Teil nun nahe der Nulllage befindet. Dieser Testlauf kann mehrmals durchgeführt werden, um die Programmfunktionalität zu überprüfen. Dabei werden die Koordinaten angezeigt und man kann sehen, dass der Winkel in dem von „PatMax“ eingestellten Bereich liegt. Hier wird auch das erste Mal der Barcode ausgegeben. Klickt man auf das Foto öffnet sich ein Fenster, in dem der Barcode als String ausgegeben wird und man dann schon testen kann, ob die gesamte Bildverarbeitung funktioniert.

Jetzt ist die Konfiguration der Bildverarbeitung und die Vorbereitung des Tasks geschehen und es kann mit der Programmierung der Programmroutine begonnen werden.

3.3 Programmierung des KRL-Programms

In diesem Abschnitt geht es um die Programmierung der Roboterbewegungen. Die Programmierung geschieht in der für KUKA entwickelten Sprache KRL (KUKA Roboter Language). Weiterhin wird aufgezeigt, wie die Toolblocks aufgerufen werden und wie eine Überprüfung des Barcodes geschehen könnte.

Die Abläufe des gesamten Programms werden anhand eines Beispielprogramms aufgezeigt und Zeile für Zeile erläutert.

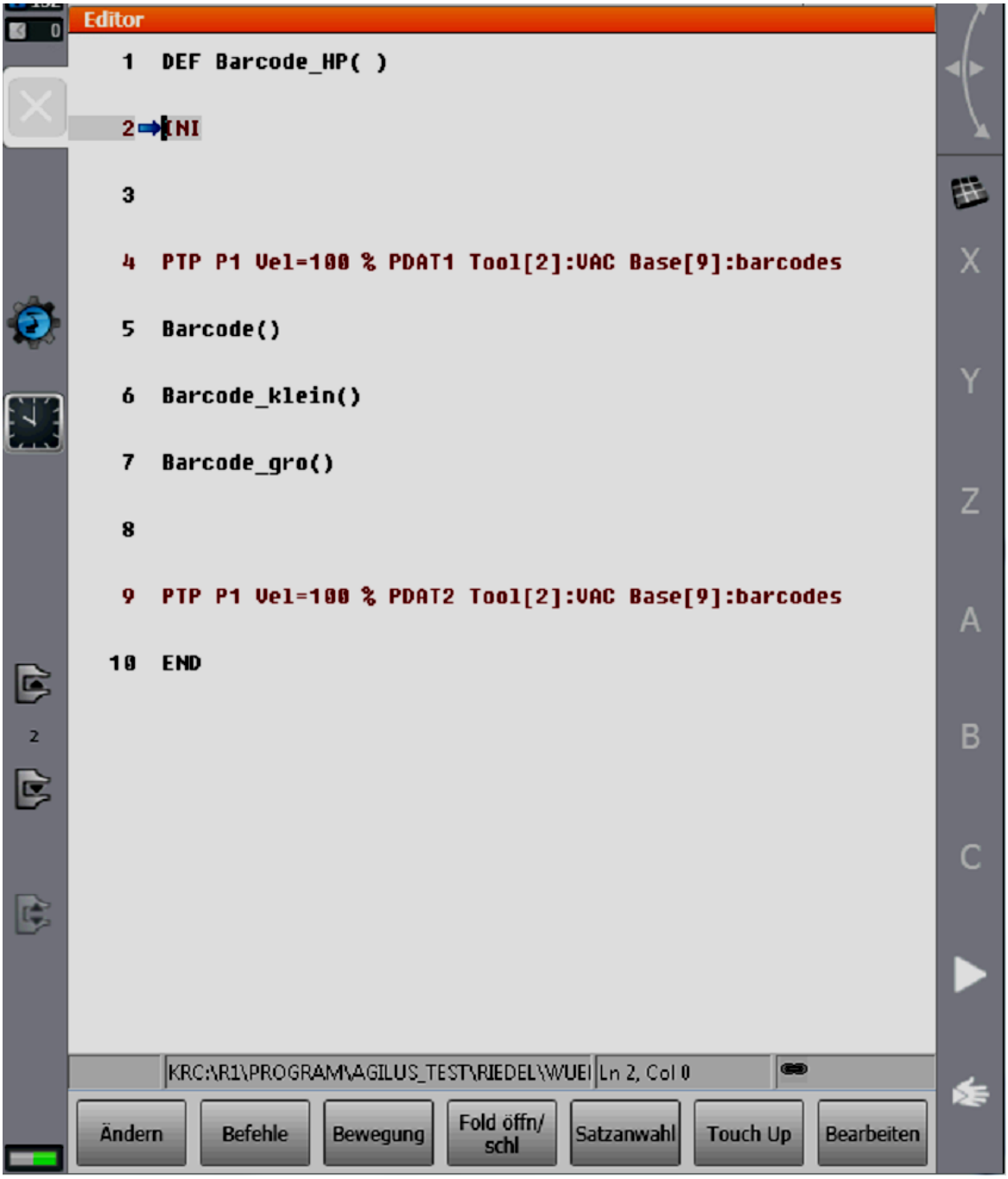

Bild 39: Hauptprogramm

Im Projekt wurden verschiedene Teile verwendet. Dafür mussten auch unterschiedliche Toolblocks angelegt werden, die nacheinander in einzelnen Programmen aufgerufen werden. Es wurde also ein Hauptprogramm erstellt, in dem die drei Toolblocks als Unterprogramme aufgerufen werden.

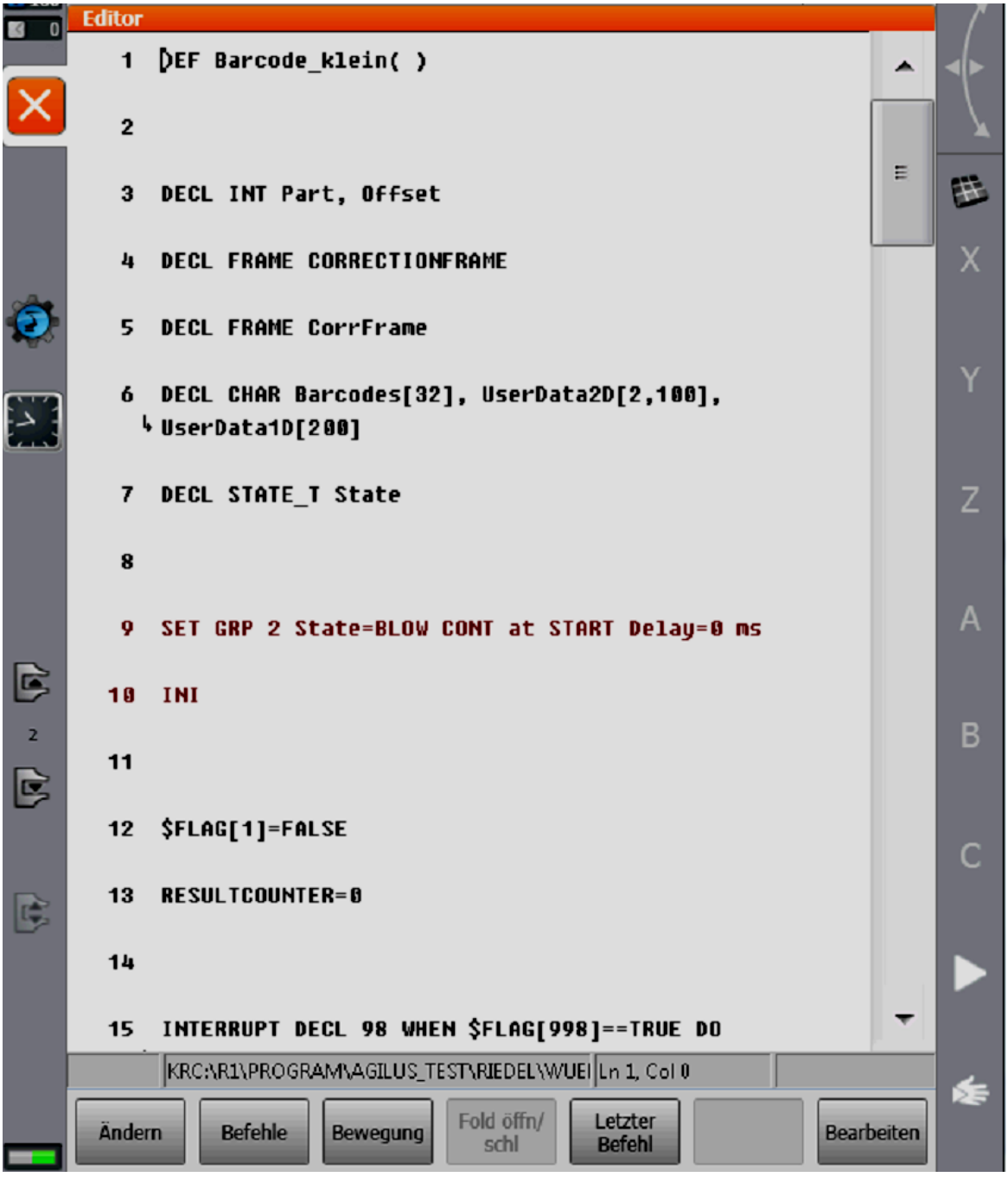

Bild 40: Programmierung 1

In **Zeile 3** werden die INT-Variablen deklariert, die für das Programm benötigt werden. Der Wert „PART“ steht für das einzelne Ergebnis einer Mustererkennung. Wird ein Objekt gefunden wird der Wert auf 1 gesetzt. Falls nicht bleibt der Wert 0 und das Programm wird beendet.

Der Wert Offset wird für die Ausgabe des Barcodes benötigt.

In **Zeile 4** und **5** werden die verschiedenen Koordinatensysteme deklariert. Der CORRECTIONFRAME ist eine spezielle KUKA-Variable und wird mit einem VT_Direct-Befehl aufgerufen. Der „CorrFrame“ beschreibt den Verschiebungsvektor zwischen dem Kamera- und dem Robotersystem.

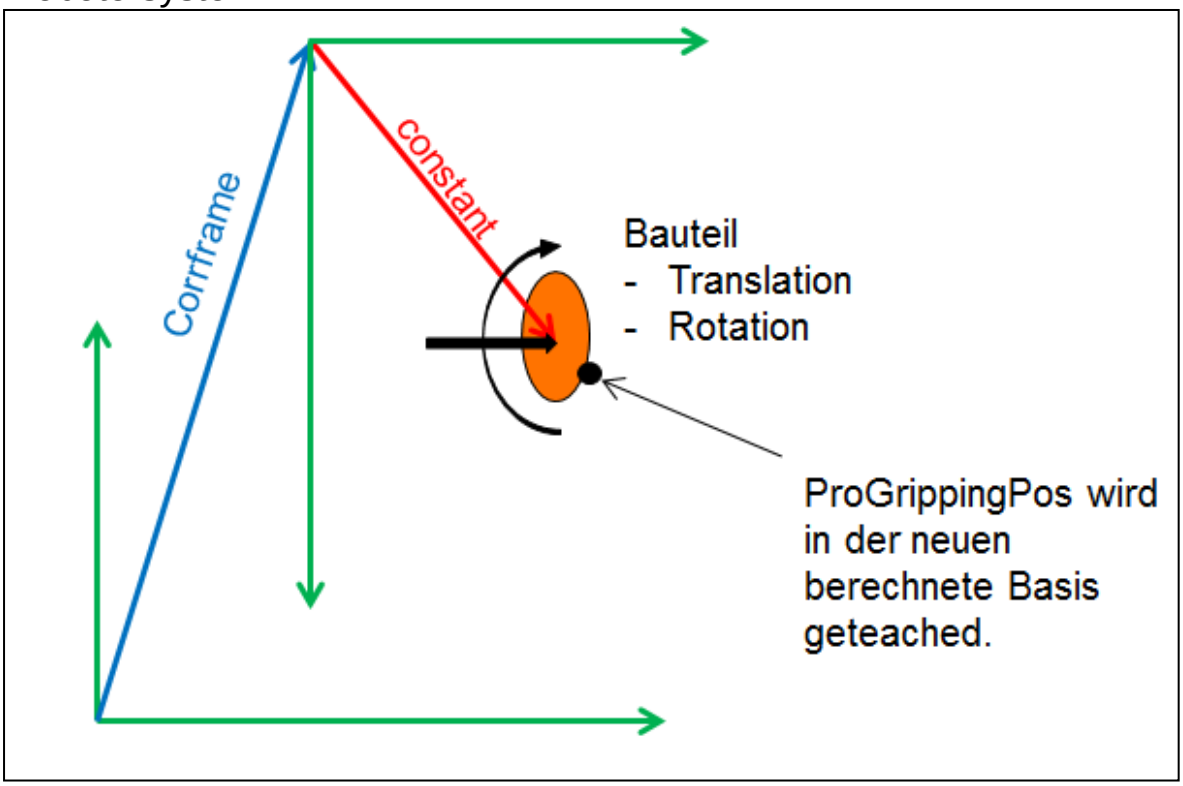

Bild 41: CorrFrame [2] Folie 59

In **Zeile 6** werden Charaktervariablen deklariert. Zuerst deklariert man die Variable „Barcodes“, welche zuvor im WorkVisual angelegt wurde, mit einem maximalen Inhalt von 32 Zeichen. Anschließend wird die UserData deklariert. Diese gibt später den Barcode als String aus und ist standardmäßig in einem zweidimensionalen Feld definiert. Die Zahlen stehen dabei für den Kameraindex und für den Inhalt. Die Zahl 2 steht also für zwei mögliche Kamerasysteme und die Zahl 100 für den Inhalt. Da im Projekt nur eine Kamera benutzt wurde, wird die zweidimensionale in eine eindimensionale Variable umgewandelt. Also wird der Inhalt von beiden Systemen in eines gepackt und ebenfalls deklariert.

Zeile 7 beschreibt einen Statustyp.

Zeile 9 dient als Sicherheitsfeature und gibt dem Saugnapf die Anweisung Luft auszublasen. Dies stammt als Merkmal aus einem Projekt

mit einem Greifer. Der Greifer wurde dabei vor Ablauf des Programms geöffnet, um nicht geschlossen in das Objekt zu fahren.

In **Zeile 12** wird das Statusflag 1 mit dem Status FALSE versehen. Das zeigt an, dass keine Daten gesendet werden.

In **Zeile 13** wird der Resultcounter gleich 0 gesetzt, so dass dieser vor dem Start zurückgesetzt wird und keine versehentlichen Daten noch im Speicher sind. Im Resultcounter wird das Ergebnis der Bildverarbeitung gespeichert. Der Wert entspricht der Anzahl der gefundenen Teile.

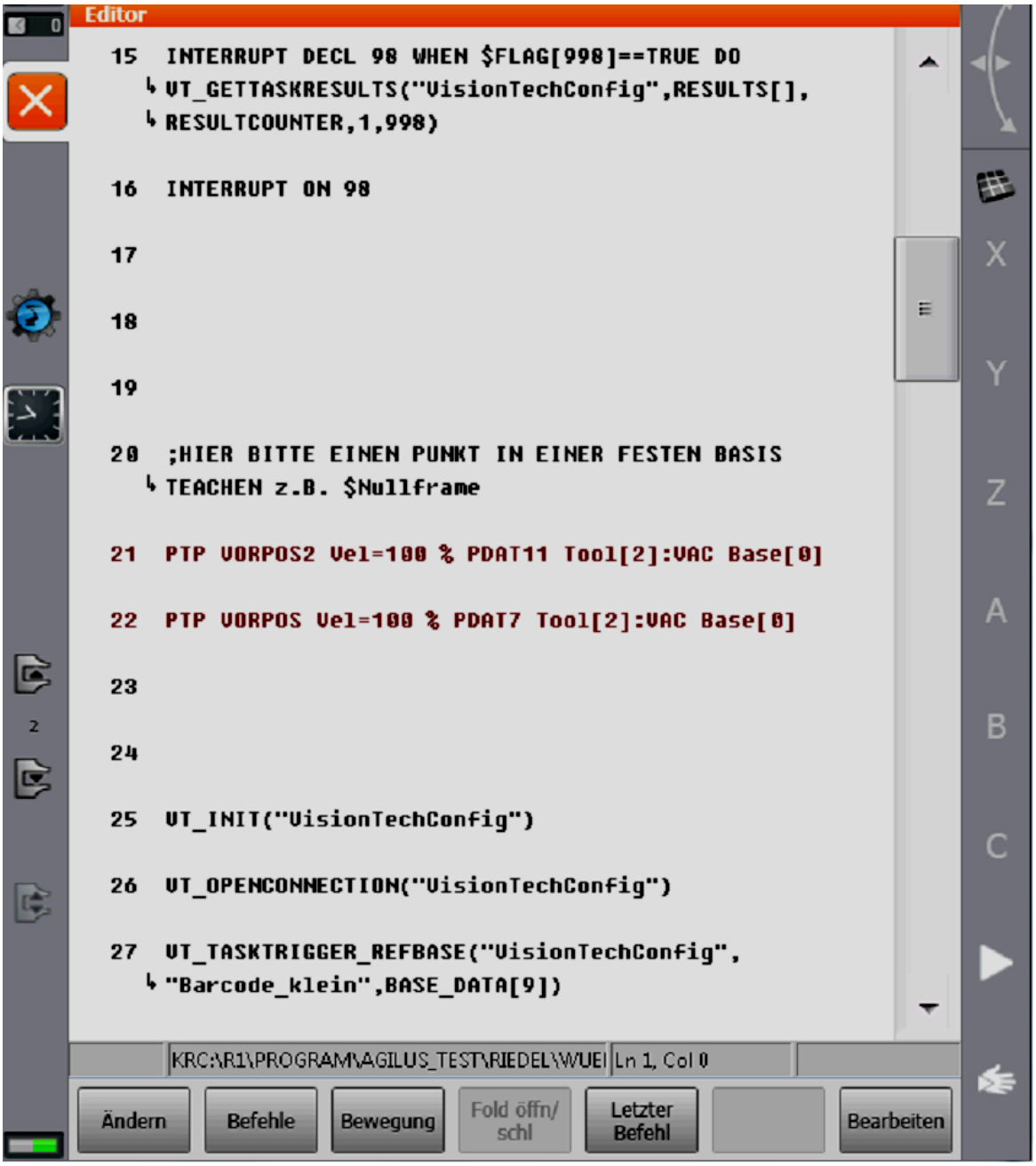

Bild 42: Programmierung 2

In **Zeile 15** steht der Interrupt-Befehl, welcher ausgeführt wird, um ein Foto vom Suchbereich zu machen. Er dient als Unterbrechung, bis das Ergebnis der Bildverarbeitung eintrifft. Das Flag 998 gibt an ob alle Daten empfangen wurden. Wenn also die Bildverarbeitung erfolgreich war und alle Daten an der Steuerung anliegen, wird der Befehl „VT_GETTASKRESULTS“ aufgerufen. Die Ergebnisse (RESULTS) werden im Resultcounter gespeichert und das Flag 1 auf TRUE gesetzt.

In den **Zeilen 21** und **22** wurden zwei Bewegungsbefehle eingefügt. Diese werden als Vorpositionen deklariert und bringt den Roboter und die Zentralhand in eine Position, von der aus die Greifposition leichter erreicht werden kann als von der HOME-Position.

Ab der **Zeile 25** wird die Bildverarbeitung gestartet. Befehle, die mit einem „VT_“ aufgerufen werden, sind VisionTech-Befehle. In Zeile 25 wird die Config-Datei aufgerufen. Danach wird die Verbindung zwischen Kamera und Steuerung, aus der „Config-Datei“ heraus, hergestellt (**Zeile 26**). In der **Zeile 27** wird nun der Trigger ausgelöst. Dazu wird aus der „Config-Datei“ der benötigte Task aufgerufen und das Ergebnis der richtigen Roboterbase zugeordnet.

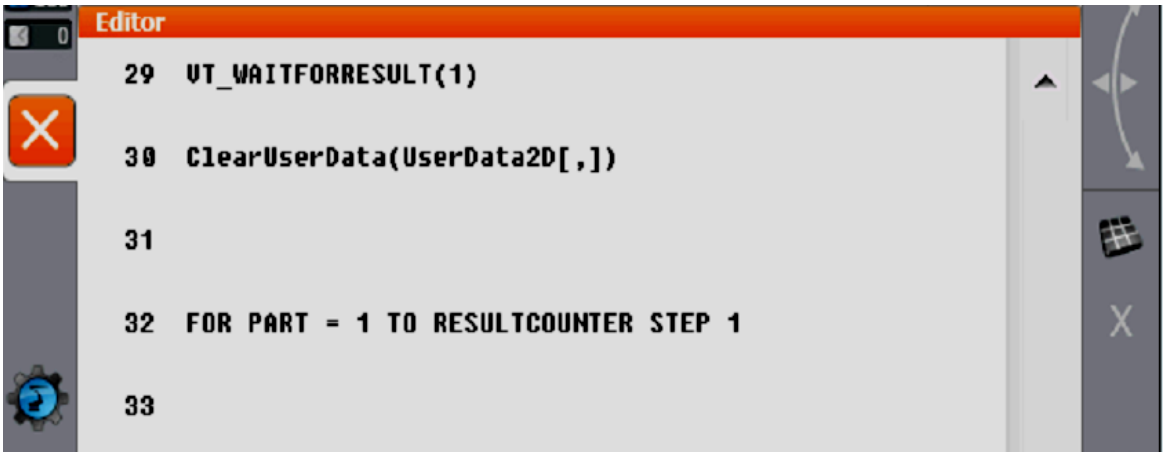

Bild 43: Programmierung 3

Wie bereits im Interrupt-Befehl erläutert werden die Ergebnisse vom Flag 1 angezeigt. Aus diesem Grund wird in **Zeile 29** darauf gewartet, dass das Flag 1 gesendet wird. Wird das Flag nicht gesendet, weil entweder kein Teil im Bildbereich liegt oder aber zu wenig Übereinstimmungsmerkmale gefunden wurden, so wird der Bildverarbeitungsprozess im KRL-Programm übersprungen und nach der FOR-Schleife fortgesetzt. **Zeile 30** dient dazu die UserData zurückzusetzen. Falls im Durchlauf zuvor Daten übermittelt und der Barcode in der UserData ausgegeben wurden, dann will man hier verhindern, dass sich diese

mit den neuen Daten überschneiden und es eventuell zu Fehlermeldungen kommt.

In den **Zeilen 32** bis **48** wird erläutert was mit den Daten passieren soll, falls welche anstehen.

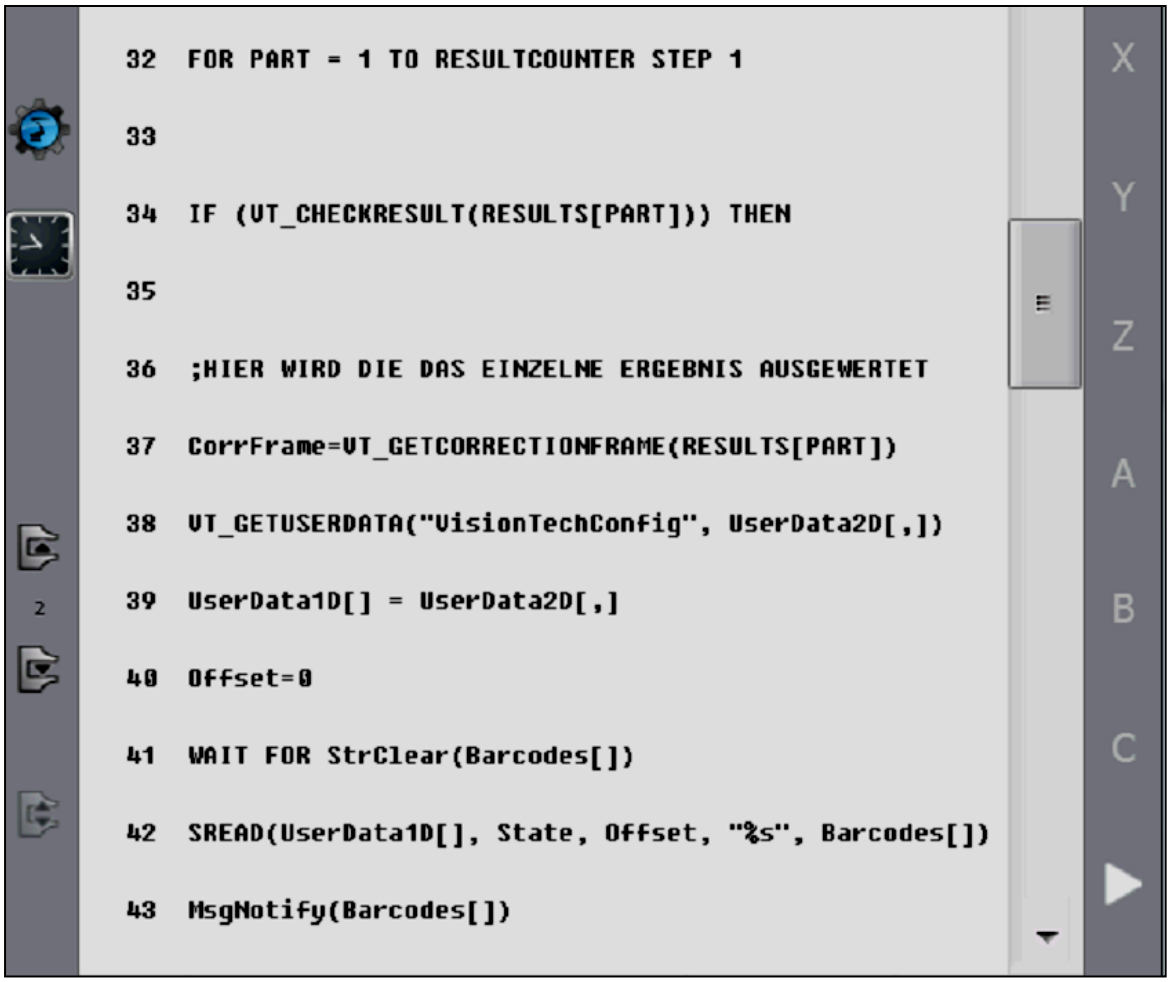

Bild 44: Programmierung 4

Wurden Bauteile gefunden und diese mit dem Flag 1 angezeigt, so wird eine FOR-Schleife aufgerufen. Wenn also ein Bauteil (PART = 1) gefunden wurde wird der Resultcounter um einen Schritt erhöht. Wenn das Ergebnis überprüft und verifiziert wurde wird danach das einzelne Ergebnis ausgewertet. Da sich die Koordinatensysteme von Kamera und Roboter in Ursprung und Ausrichtung unterscheiden können wird hier ein CorrectionFrame eingefügt. Dieser ist eine Korrektur und soll diesen Versatz beschreiben. Anschließend wird die UserData bearbeitet. Hier wird zuerst die zweidimensionale Struktur in eine eindimensionale geändert und gleichgesetzt. Somit wird der Barcode von nun an in die 1D-Struktur geschrieben. Danach wird der Offsetwert auf null gesetzt. Der Offset wird hier nur von der Funktion SREAD verwendet. Er gibt an, an welcher Stelle ein Buchstabe ausgelesen wird und er-

höht sich während des Auslesens immer automatisch um eine Position je Buchstabe. Das bedeutet, wenn das Wort z.B. „ABCD“ heißt und mit SREAD ausgelesen wird, muss der Offset vor dem Befehl auf null gesetzt werden, damit beim „A“ begonnen wird. Der SREAD-Befehl zählt dann immer eine Stelle nach oben bis er am Ende des Strings angekommen ist. Somit sollte der Offset nach verrichteter Arbeit auf 4 stehen. Vor dem nächsten Durchlauf wird daher der Offset wieder zurückgesetzt, damit wieder beim ersten Buchstaben und nicht beim vermeintlich fünften begonnen wird.

Der Barcode wird als String in eine Liste geschrieben. Um wieder den Fehlermeldungen vorzubeugen wird vorher der String zurückgesetzt. Wenn dies geschehen ist wird der neue String nun mit einem SREAD-Befehl herausgelesen. „Übersetzt“ bedeutet **Zeile 42**: Lese aus der eindimensionalen UserData den Status und den Offset heraus sowie den String namens „Barcodes“. Die Variable „State“ ist vom Typ „State_T“ und muss ebenfalls deklariert werden (siehe Zeile 8). Der Befehl zum Auslesen von Strings verwendet diesen Status zur Fehlerauswertung. Das bedeutet in „State“ werden automatisch Informationen abgelegt, ob der Befehl korrekt ausgeführt wurde. Anschließend wird der String über den „MsgNotify“-Befehl ausgegeben. Die Funktion „MsgNotify“ ist eine Standard KUKA-Funktion, um einfach und schnell eine Hinweismeldung zu generieren. Der Parameter „Barcodes[]“ ist hierbei ein Array aus Characters (also ein String), der dann in der Meldung ausgegeben werden soll. Damit wird der gesamte Inhalt des aus der Bildverarbeitung kommendenn Strings „UserData“ in die Variable „Barcodes[]“ umgespeichert. Falls die Informationen geändert werden müssen, die „Barcodes[]“ enthält, dann muss im WorkVisual alles, was in den Toolblockausgang „UserData“ übergeben wird auch geändert werden.

Der Barcode wird nun im Benachrichtigungsfeld angezeigt.

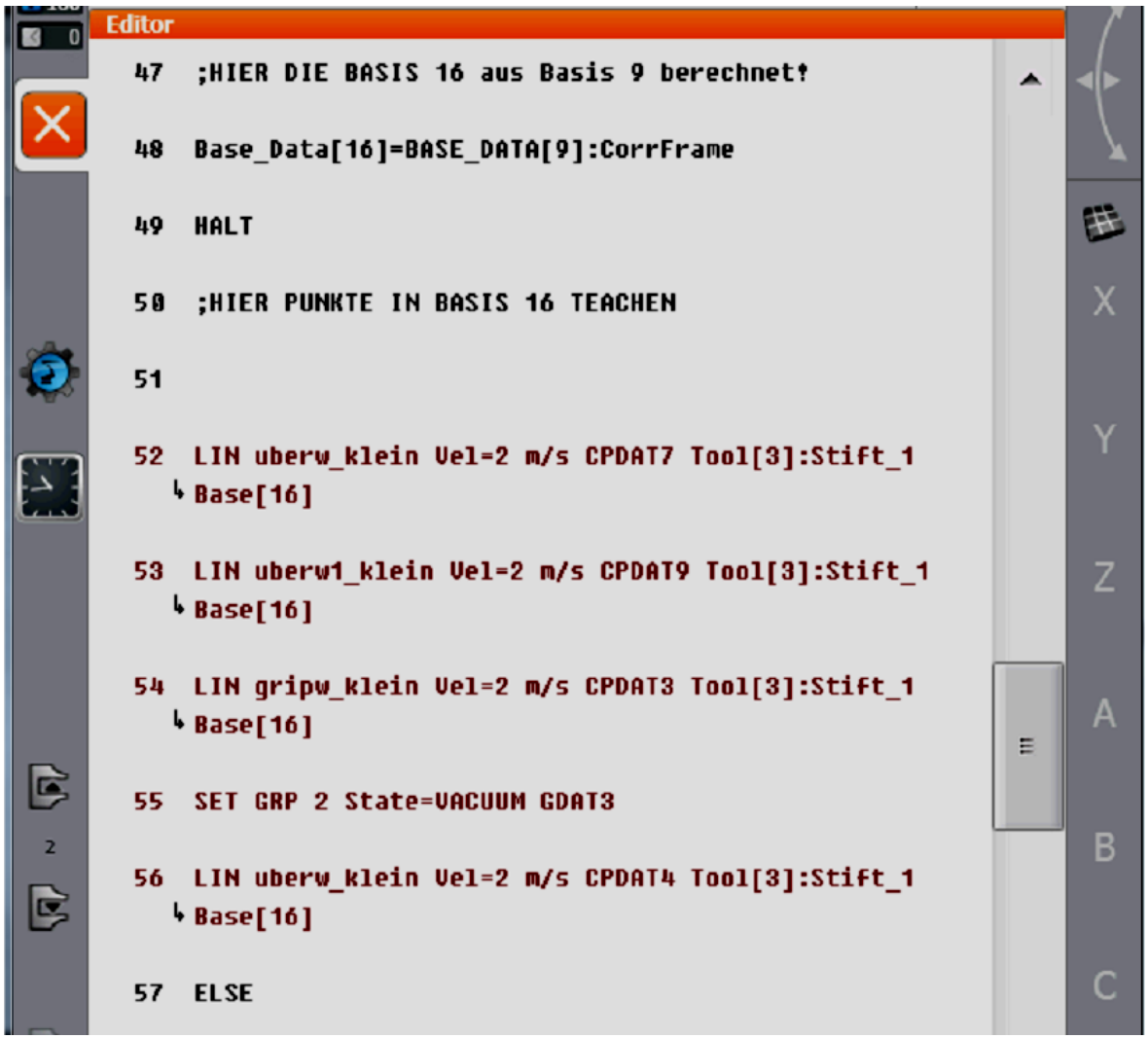

Bild 45: Programmierung 5

Im Anschluss an die Barcodeausgabe wird das Objekt nun angefahren. Da jedoch weder die Basis 9 noch der *CorrectionFrame* „berechtigt“ sind das Objekt richtig anzufahren und zu greifen muss eine neue unabhängige Basis benutzt werden. Der *CorrectionFrame*, wo die Roboter- und die Kamerabasis vereint wurden, wird nun in die Basis 16 temporär eingetragen. In **Zeile 48** wird die Basis 9 mit dem *CorrFrame* der Basis 16 gleichgesetzt. Ab **Zeile 52** werden Bewegungsbefehle geschrieben. Dabei soll der Roboter linear über das Objekt fahren, danach linear nach unten (*uberw1_klein*) und anschließend die Greifposition anfahren. Das Teil wird mit dem SET-Befehl angesaugt und nach oben bewegt.

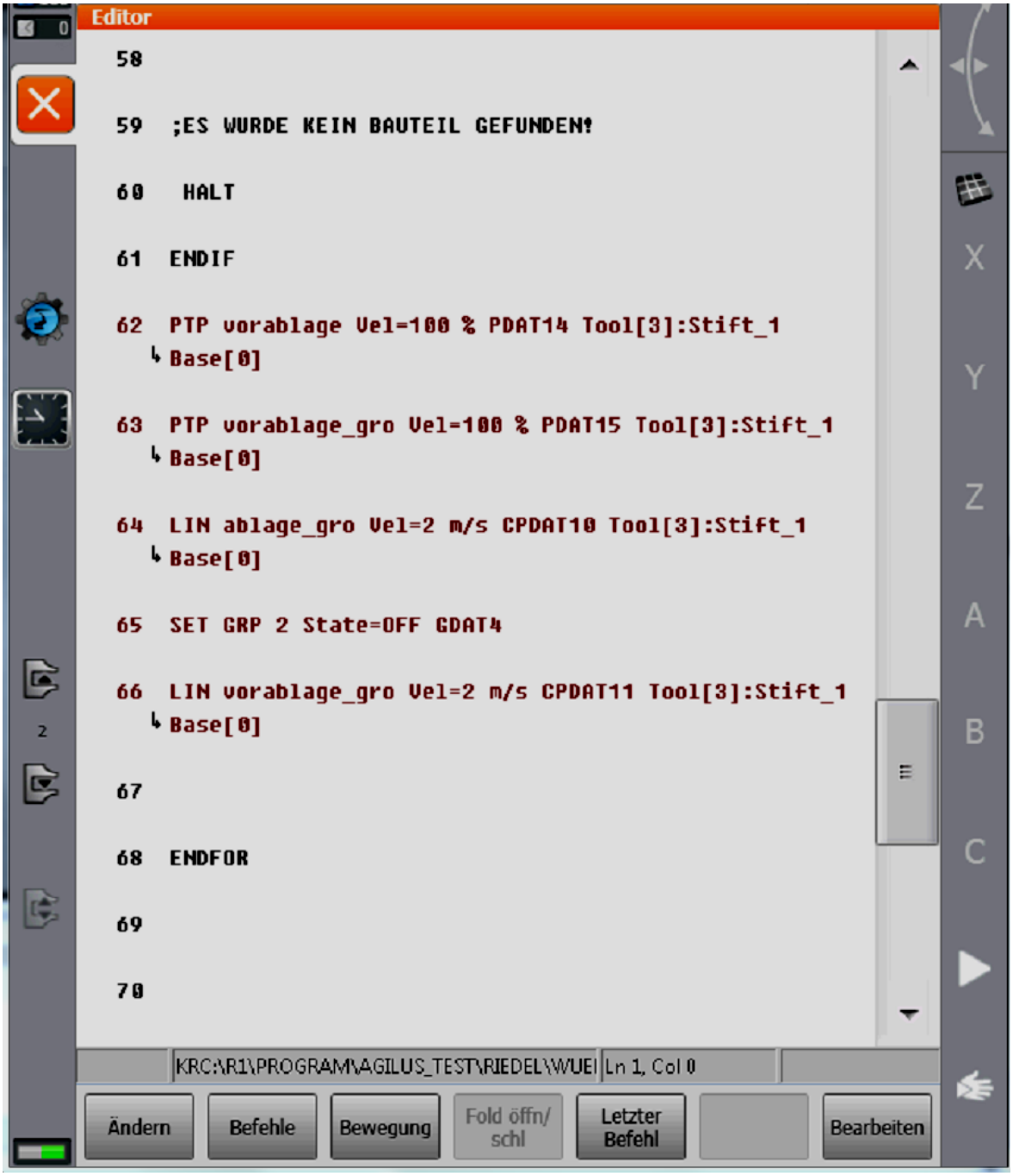

Bild 46: Programmierung 6

In **Zeile 61** wird die IF-Anweisung geschlossen, die zuvor geöffnet wurde, weil ein Objekt erkannt wurde (**Zeile 34**). Danach wird das Teil zur Ablageposition gebracht und mit dem BLOW-Befehl abgelegt. Die FOR-Schleife wird ebenfalls geschlossen.

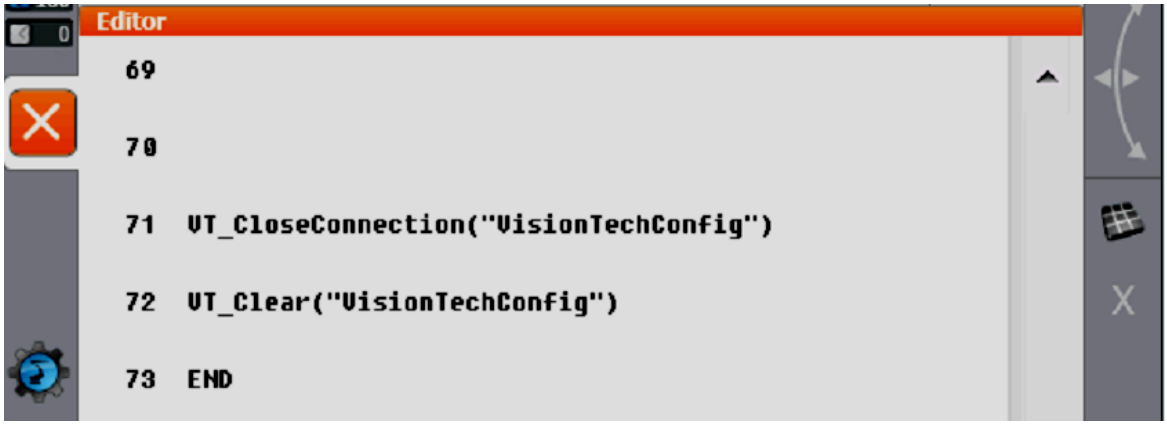

Bild 47: Programmierung 7

Zum Schluss wird die Verbindung zur Kamera beendet (**Zeile 71**) und die Config-Datei zurückgesetzt.

In **Zeile 30** wird die UserData auf den Empfang der Daten vorbereitet, indem der Inhalt gelöscht wird. Folgendes Bild soll erläutern wie man diese Funktion deklariert und beschreiben kann.

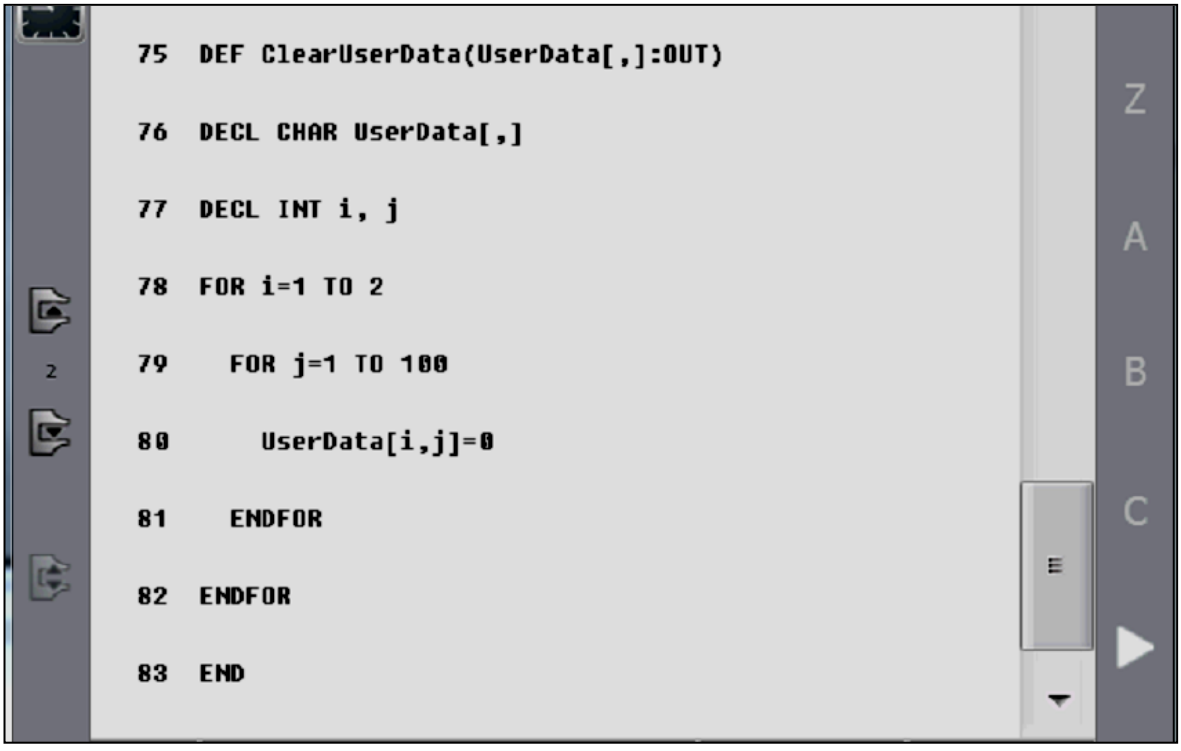

Bild 48: ClearUserData

Die UserData wird in **Zeile 75** definiert und als auszugebendes Element hinzugefügt. Anschließend wird dafür der Typ Charakter vergeben. Da die Umwandlung der zweidimensionalen in eine eindimensionale UserData noch nicht im Programm geschehen ist, müssen hier beide Elemente der UserData betrachtet werden.

Die Elemente beziehungsweise der Inhalt werden als Ganzzahltyp (INT) deklariert (**Zeile 77**). In den **Zeilen 78** bis **80** werden die einzelnen Elemente solang mit 0 aufgefüllt bis in der UserData keine Daten mehr enthalten sind. „Übersetzt“ heißt diese Anweisung: „Für den Fall, dass in der Variable i ein Inhalt steht, so fülle diesen bis 2 mit Null auf. Für den Fall, dass in j ein Inhalt steht, so fülle diese bis 100 mit Null auf“. Somit werden die Inhalte der UserData gelöscht bis der Wert Null in ihr steht. Danach werden die FOR-Schleifen beendet.

Nun ist es möglich ein Objekt zu greifen und einen Barcode auszugeben. Dabei eröffnen sich noch Möglichkeiten dieses Programm zu verfeinern. So könnte man zum Beispiel die Ablageposition anhand des Barcodes beeinflussen. Wird also ein Objekt erkannt und gegriffen, so soll der Barcode ausgelesen und daraufhin die Ablageposition gewählt werden. Dies wurde mit folgenden Zeilen durchgeführt.

```
6  DECL CHAR Barcodes[32], UserData2D[2,100],
  ↳UserData1D[200]

7  BOOL B

8  DECL STATE_T State

9

10➡Barcodes[]="568754856745"

11
```

Bild 49: Strings vergleichen 1

Hierzu wird eine neue Variable angelegt. Diese Variable soll anschließend mit einem festen Wert verglichen werden. Da hierbei nur ein TRUE- oder FALSE-Ergebnis erwartet wird, wurde als Variablentyp für die Variable „B“ der Typ BOOL gewählt. Danach wird eine konstante Variable „Barcodes“ vereinbart. Diese wird später verglichen. Da im Programm keine anderen Barcodes erwartet werden ist dieses Beispiel zwar sehr einfach, verdeutlicht aber, wie ein Vergleich zwischen verschiedenen Strings funktioniert.

In **Zeile 65** wird über den Befehl *StrComp* zuerst die Variable herausgelesen, mit der Konstante verglichen und auf ein Ergebnis in der Form TRUE oder FALSE erwartet. Die Anweisung „#NOT_CASE_SENSE“ bedeutet, dass die Groß- und Kleinschreibung nicht beachtet wird.

```
64

65  B = StrComp(Barcodes[], "568754856745",
   ↳ #NOT_CASE_SENS)
```

Bild 50: Strings vergleichen 2

Somit ist das gesamte Programm funktionsbereit. Das Programm kann so gespeichert werden und über den Button **Anwählen** wird die Steuerung aktiviert. Über das Bedienpanel lässt sich das Programm im besten Fall gleich ohne Fehlermeldungen starten. Dabei kann zwischen dem automatischen Betrieb, dem Schritt-für-Schritt-Betrieb und dem Bewegungsbefehl-Betrieb unterschieden werden. Der Unterschied liegt darin, dass man zuerst alle Bewegungsbefehle und im Schritt-für-Schritt-Betrieb jeden einzelnen Befehl bestätigen muss.

Als weiteres Beispielprogramm kann die Dokumentation von KUKA aus den Schulungsunterlagen [4] genutzt werden. Diese liegt im Labor aus oder kann von KUKA angefordert werden.

4 Problembehandlung

Während der Arbeiten traten immer wieder Probleme auf, die teilweise größere Verzögerungen nach sich zogen, angefangen von der einfachen Suche nach einem benötigten Ordner bis hin zur inhaltlichen Fehlersuche, wenn das Programm nicht funktionierte. Dieser Abschnitt soll ein paar dieser Fehler erklären und Aufschluss darüber geben, in welchem Ordner bestimmte und häufig verwendete Dateien liegen.

4.1 Fehlermeldungen und Lösungsansätze

- Variable nicht deklariert

Neue Variablen müssen entweder in der src.- oder in der dat.-Datei deklariert werden. Außerdem muss auf Schreibfehler geachtet werden.

- Doppeldeklaration

Variablen werden fälschlicherweise in der dat.-Datei und in der src.-Datei deklariert. Weiterhin ist es möglich, dass eine Variable im Laufe des Programms falsch benutzt oder anders definiert wird.

- Muster werden im WorkVisual nicht erkannt

Dies kann mehrere Ursachen haben. Zu allererst sollten die Trainingsbilder von sehr hoher Qualität sein. Außerdem sollten die Lichtverhältnisse möglichst konstant sein. Man sollte probieren, wie sich das Ergebnis verhält, wenn man die Güte herabsetzt. Werden die Kanten nicht erkannt, so sollte der Kontrastschwellwert im „PatMax“ herabgesetzt werden. Im Reiter „Grafiken“ können bestimmte Ergebnisse angezeigt werden. Hier kann man abwechselnd die Anzeige „fein“ und „grob“ ausschalten. Werden aufgrund dessen im Ergebnis spezielle Konturen auf der Oberfläche angezeigt kann man davon ausgehen, dass diese der Hauptgrund für die Nichterkennung sind. Dazu streicht man diese Konturen im Maskeneditor heraus und der Schwerpunkt wird nun auf die Außenkanten gelegt.

- Barcode wird nicht erkannt

Hier liegt meist der Fehler in den Verzweigungen, die den Ablauf und die Ausgabe beeinflussen. Weiterhin sollte überprüft werden, ob der richtige Barcodetyp ausgewählt wurde. Im Projekt selbst trat der Fehler auf und es stellte sich heraus, dass der Barcode zu knapp abgeschnitten wurde und somit die Kante des Papiers als Element des Codes erkannt wurde. Somit passte der Code nicht mehr zum eingestellten Typ.

- Teil wird nicht oder falsch angefahren

Dafür kommen mehrere Möglichkeiten in Betracht, die nun erläutert werden.

1. **Kalibrierverwaltung** anklicken und eigenes Projekt wählen sowie Kalibrierung wählen.
2. Legen Sie das Bauteil mittig ins Kamerabild, das Bauteil danach nicht bewegen. Erzeugen Sie einen Modellframe für ihren Task. Starten Sie das KRL-Programm und fahren Sie bis die „VT_WaitForResult(1)“ Zeile abgearbeitet ist. Öffnen Sie die Variablenzeige (Anzeige => Variable => Einzeln), und geben Sie als Name "Results[]" ein. Das Ergebnis sollte momentan sehr kleine Werte für X,Y,A zeigen (alles sehr nah an 0).
3. Teachen Sie bitte die Punkte neu, die über dem Muster liegen, und den Greifpunkt (In Base_Data[16]) mit Werkzeug 1. Setzen Sie das Programm zurück und fahren Sie das Bauteil an (immer noch nicht bewegen). Der Roboter sollte an das Bauteil fahren!
4. Bewegen Sie das Bauteil entlang einer Richtung ohne es zu verdrehen. Setzen Sie Ihr Programm zurück und fahren das Bauteil an. Der Roboter sollte an das Bauteil fahren! (Fährt der Roboter zu weit oder zu kurz, aber in die korrekte Richtung wurde eine falsche Kalibrierplatte während des Kalibrierens ausgewählt.)
5. Rotieren Sie das Bauteil nun an der verschobenen Position und setzen Sie das Programm zurück und fahren das Bauteil an. Der Roboter sollte nun an das Bauteil fahren!

Wenn der Roboter nun nicht mehr an das Bauteil fährt, wurde vermutlich während des Kalibriervorgangs mit dem Roboter ein Koordinatensystem vermessen, welches zwar parallele Achsen zum Kamerakoordinatensystem besitzt, jedoch nicht denselben Ursprung hat.

Wurden verschiedene Objekte mit der gleichen Kalibrierebene gesucht können Fehler auftreten, wenn sich die Merkmalhöhe (1/2 Kugelhöhe (maximaler Durchmesser) und Würfelhöhe(Oberkante des Würfels)) unterscheidet. Deswegen empfiehlt es sich generell für neue Objekte auch eine neue Kalibrierebene anzulegen.

- Teil wird solange angefahren bis es rotiert wird

Sie haben, nachdem der Task eingerichtet wurde, die Kalibrierung verändert, aber im Task nicht die neue Kalibrierung ausgewählt.

Der TCP des Werkzeugs während des Kalibrierens liegt weit entfernt vom realen TCP.

Wahrscheinlichster Fall: Während der Kalibrierung wurde nicht der korrekte Ursprung auf der Platte angefahren. Dazu noch ein Ablaufschema.

1. Wählen Sie ihre Messspitze als Werkzeug an.

2. Setzen Sie das Werkzeug für die Funktionstasten.

3. Rotieren Sie mit dem TCP um die A-, B- und C-Winkel und prüfen Sie, dass die Spitze ihres Werkzeugs die Position nicht verändert (im Idealfall << 1mm).
Wenn dies nicht der Fall ist vermessen sie Ihr Werkzeug neu.

4. Legen Sie die Kalibrierplatte unter die Kamera und führen Sie die Kalibrierung neu durch! Bitte geben Sie während der Kalibrierung auch einen neuen Namen für die Kalibrierung an! Legen Sie besonderen Wert auf diesen Punkt, da (falls die Messspitze nicht mehrere Millimeter außermittig zum „Tool_Data“ gesessen hat) hier der Fehler im Ablauf liegt! Das Fiducial muss dabei beachtet werden. Der Ursprung des Koordinatensystems liegt dabei im verlängerten Schnittpunkt des Fiducial! Dieser Endpunkt ist der erste Punkt der bei der Drei-Punkt Kalibrierung eingemessen wird!

5. Die Basis auf der Kalibrierplatte sollte man auch einmessen und anschließend zurück ins VisionTech-Plug-In und speichern!

6. Nun den eigenen Task anwählen, auf das Kamerasymbol klicken und die geänderte Kalibrierung mit dem Task aktualisieren.

7. Bauteil zentrisch unter die Kamera legen und ein Modell erzeugen.

8. KRL-Programm soweit durchfahren, bis die Basis berechnet wurde.

9. Die Punkte neu teachen.

10. Bauteilposition beibehalten, Programm zurücksetzen und beobachten, ob das Bauteil richtig angefahren wird.

11. Bauteil in X und/oder Y verschieben, aber nicht drehen, Programm zurücksetzen und anfahren.

12. Bauteil nun rotieren und versuchen anzufahren.

Das Problem sollte nun behoben sein. Ansonsten sollte der Support vom Hersteller kontaktiert werden.

4.2 Überblick über die Dateipfade

Änderung der Config-Datei

C:\KRC\Roboter\Config\User\Common\EthernetKRL

Bilder der Kalibrierung auf Seite der Steuerung

C:\KRC\TP\VisionTech\Snapshots

Toolblocks auf Seite der Steuerung

C:\KRC\TP\VisionTech\Toolblocks

Bilder aus WorkVisual

C:\KUKA\Ordner

„KRCDiag“-Datei im VisionTech

Datei→Archivieren→USB (Schrank) →KRCDiag

Bevor die „KRCDiag“ ausgelesen werden kann muss ein USB-Stick an die Steuerung angeschlossen werden. Die Datei kann an den Hersteller geschickt werden, um Fehler zu finden. In der „KRCDiag“-Datei sind nur Daten der Steuerung enthalten und keine Toolblocks.

5 Fazit und Ausblick

Zusamenfassend ist zu sagen, dass es nun möglich ist im Labor den Einsatz von Kamerasensoren näher zu untersuchen und die Funktionsweise verschiedener Tools und Bildbearbeitungen kennenzulernen. Es wurden Positionsbestimmungen und eine Barcodeerkennung durchgeführt und miteinander verglichen.

Im Vergleich zwischen dem Arbeiten mit der neuen KRC4-Steuerung in Verbindung mit dem WorkVisual und dem InSight-Explorer der KR3-Steuerung lässt sich sagen, dass es einen großen Unterschied gibt. Während im InSight-Explorer die komplette Bildverarbeitung direkt an das Programm angebunden war ist diese in der KRC4-Steuerung durch VisionTech und WorkVisual ausgegliedert worden. In einer Praktikumsarbeit [3] wurde der Einsatz verschiedener Bildverarbeitungstools untersucht und man kann so die Unterschiede zwischen beiden Programmen verdeutlichen. Vorteilhaft im InSight war, dass man sämtliche Einstellungen und Ergebnisse schnell auf dem Desktop hatte und so relativ unkompliziert Änderungen vornehmen konnte. Weiterhin konnte man im InSight einzelne Codes einfügen und Berechnungen ausführen, während man die Bildverarbeitung Im VisionTech und WorkVisual durchblicken musste und lernen musste welches Programm welchen Teil übernimmt. Der Vorteil an dieser Methode ist, dass es verschiedene Bereiche gibt, die in sich geschlossen, vom KRL-Programm abgekoppelt und somit auch geschützt sind. Der Nachteil daran ist, falls Änderungen getroffen werden müssen, so ist es im schlechtesten Fall nötig, eventuell einen neuen Task zu erstellen, einen neuen Toolblock anzulegen, andere Bilder aufzunehmen und die Kalibrierung anzupassen. Dennoch kann man die neue Methode der Bildverarbeitung als besser betrachten, da auch zusätzliche und weiterentwickelte Tools zur Verfügung stehen.

Ob sie nun als einfache Anwesenheitssensoren oder zur genauen Positionsbestimmung von Körpern eingesetzt werden, Sensoren wie Kameras gewinnen immer mehr Zuspruch und Bedeutung in Forschung, Entwicklung und Produktion. Die Entwicklung scheint dabei noch nicht zu Ende zu sein. So werden voraussichtlich die Gehäusekörper immer kleiner und die Sensoren sowie deren Übertragungsraten immer größer.

Dies dürfte sich zwar auf die momentan schon relativ hohen Kosten auswirken, welche jedoch auch bei einer hohen Produktion und Nachfrage wieder sinken könnten, zumal die Hersteller oft eine Testphase anbieten, um hier eine bessere Integration für einen schnelleren Einsatz zu ermöglichen.

Das Thema der Beleuchtung ist immer noch eine wichtige Baustelle. Durch sich stark ändernde Lichtverhältnisse können vielleicht einige Algorithmen und Funktionen nicht einwandfrei ausgeführt werden bzw. müssen manchmal die Ergebnisse angezweifelt werden. Zwar wurde dieses Problem bereits durch die automatische Belichtung des „PatMax“–Algorithmus angegangen. Da sich die Kamera jedoch unweit von einem großen Fenster befindet ist diese eventuell nicht immer so effektiv wie erhofft. Dies spiegelt sich nicht nur in der Trefferquote und der Güte, sondern auch in der genauen Klassifizierung der Objekte wider. In den Anlagen (A-I, A-II) finden sich einige Tabellen, die verdeutlichen, welchen Einfluss die Beleuchtung hat. Vielleicht sollte man aus den eben genannten Problemen über neue Methoden der Bilderkennung nachdenken. Wenn das Licht so viele Probleme bereiten kann, ist es dann nicht sinnvoll über Infrarot oder Ultraschall nachzudenken? Somit würden sich möglicherweise Vorteile ergeben. Zum einen können keine Schatten mehr die Ergebnisse beeinflussen. Daraus ergibt sich, dass man vielleicht nur noch eine Kalibrierebene braucht, da die Höhe des Objektes irrelevant wäre. Zum anderen müssten Teile nicht mehr bearbeitet werden, um Kanten zu erkennen. Kontrastschwellwerte und Helligkeitseinstellungen müssten somit nicht mehr empirisch ermittelt werden, sondern würden möglicherweise ganz aus der Software gestrichen werden können. Nachteilig wäre eventuell, dass Barcodes oder Schriften nicht mehr erkannt werden können. Je nach Einsatzgebiet der Kamera ist aber auch das möglicherweise zu vernachlässigen. Die Entwicklung ist also noch nicht abgeschlossen und es bleibt abzuwarten welche technischen Weiterentwicklungen in Zukunft die Produktion erleichtern.

Literatur

[1] < http://www.kuka-robotics.com/germany/de/company/group/kuka_ag/ >, verfügbar am <11.02.2016, 09:34 Uhr>

[2] <VisionTech_Collegeschulung_Q32015_inkl_Lageplan.pdf>, <2015>

[3] <KST_VisionTech_30_de.pdf>,<verfügbar am 11.04.2016>

[4] <Riedel>, <Fabian>: <Praktikumsbeleg: Experimentierplatz eines Visionsystems>,
<Mittweida>, <2015>

Anlagen

Anlagen, Teil 1

Lichtkranz, starker Sonnenschein, Deckenleuchte		
automatische Kontrastschwelle bei 5 Akzeptanzschwelle bei 60%		
Messreihe	Güte	
1	98,81	
2	97,99	
3	98,23	
4	96,22	
5	98,88	5 benötigte Versuche

Tab. 1

Lichtkranz, bewölkt, Deckenleuchte		
automatische Kontrastschwelle bei 8 Akzeptanzschwelle bei 60%		
Messreihe	Güte	
1	96,12	
2	90,15	
3	97,55	
4	94,63	
5	97,61	5 benötigte Versuche

Tab. 2

starker Sonnenschein

automatische Kontrastschwelle bei 14
Akzeptanzschwelle bei 60%

Messreihe	Güte
1	85,23
2	98,45
3	86,44
4	77,99
5	94,4

7 benötigte Versuche

Tab. 3

keinerlei Beleuchtung

automatische Kontrastschwelle bei 18
Akzeptanzschwelle bei 60%

Messreihe	Güte
1	85,45
2	86,45
3	66,12
4	84,12
5	89,12

6 benötigte Versuche

Tab. 4

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 2. Mai 2016

_____

Fabian Riedel