Studienarbeit.docx:

kann wiki ?

设备结构： 硬件：anlage台子，软件：vscode extension等等

Abkürzungsverzeichnis:

bzw.

ROI:

IDL:

DDS: Data Distribution Service

z. B.

d. h.

ca.

1. aufgabenstellung

2.bewegung

3.opencv

要校准的是哪个圆，给个simulation里的圆孔截图（仰视图）

参数啥的都用特殊字体

|  |  |
| --- | --- |
| **Abschnitt** | **Element** |
| **Vorspann** | Deckblatt |
| Aufgabenstellung |
| ggf. Sperrvermerk |
| eidesstattliche Versicherung |
| Kurzzusammenfassung englischer Abstract |
| **Verzeichnisse** | Inhaltsverzeichnis |
| Abbildungsverzeichnis |
| Tabellenverzeichnis |
| Abkürzungsverzeichnis |
| Symbolverzeichnis |
| **Textteil** | Einleitung |
| Theorieteil |
| Hauptteil |
| Zusammenfassung |
| Ausblick |
| **Nachspann** | Literaturverzeichnis |
| Anhang |

## 

**1. einleitung**

**2. aufbau und theorie**

下半平面是圆，有缺口。本文是用检测椭圆来检测圆的。

**1. Einleitung**

**分三段**

In der Präzisionsmontage ist die Entwicklung für neue Varianten oder Produkte immer aufwändig. Der Grund liegt daran, dass das Merkmal der Präzisionsmontage eine hohe Variantenvielfalt bei geringen Stückzahlen. Dabei werden die Qualität und Genauigkeit ebenfalls anspruchsvoll angefordert. Das führt zu einem sehr hohen Programmieraufwand neben der Entwicklung der entsprechenden Hardware für die Montageprozesse.

Um die daraus resultierenden hohen Stillstandzeiten der Maschine zu vermeiden, wird der gesamte Montageprozess in einer Simulationsumgebung ausgeführt. Dazu wurde bereits ein Simulationsmodell der Match- Präzisionsmontagezelle in Gazebo implementiert. Aber wenn ein Montageprozess in der Simulationsumgebung durchgeführt wird, muss das Simulationsmodell auf das reale System angepasst. Andernfalls hat es eine Abweichung zwischen der Simulation und realität, bzw. einen Real2Sim-Gap zur Folge.

Ziel dieser Arbeit ist die Verringerung dieses Real2Sim-Gaps eines Greifers(...) durch die automatische Kalibrierung der Simulation. Aus technischen Gründen wird der Greifer durch die Schrauben montiert, was zu eine kleine Abweichung der Position des Greifers führt. Da das aktuelle Simulationsmodell der Präzisionsmontageanlage auf dem CAD-Modell basiert, stimmen die Posen (des Greifers?) nicht mit den realen Posen überein. Es wird daher angefordert, dass die Position des Greifers in dem Modell kalibriert wird, um die Simulation bestmöglich der Realität anzupassen. Die Kalibrierung wird auf der Basis der Open-Source-Computer-Vision-Bibliothek, kurz OpenCV durchgeführt. Außerdem wird die Steuerung des Roboters mit der Hilfe von ROS erreicht. Zur Vermessung des… wird auch die Unterseitenkamera der Präzisionsmontageanlage eingesetzt.

Die Hauputaufgaben in dieser Arbeit besteht aus zwei Teilen: Steuern… und Computer Vision mit OpenCV 。。。

Zunächst wird in dieser Arbeit...

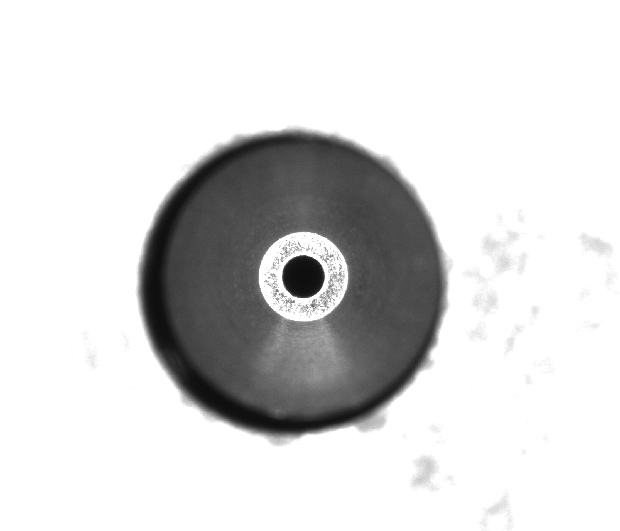
2. Bestandteile(hardware) und Theorie

2.1 Real2Sim-Gap

Heutzutage werden viele Entwicklungsaufgaben in der Simulationsumgebung durchgeführt. Die Verwendung von Simulationsumgebungen senkt nicht nur die Entwicklungskosten, sondern verbessert auch die Test- und Entwicklungszyklen und bietet die Entwicklung eine hohe Flexibilität[[1]](#footnote-2). Aber eine Simulation, die nicht gut mit der Realität übereinstimmt, führt zu dem Mißerfolg der Aufgabe. Der Grund dafür ist die Existenz der lücken zwischen der Realität und Simulation, bzw. der Real2Sim-Gap oder der Sim2Real-Gap. Der Gap wird verursacht durch z. B. das Licht, die Vibration usw. in der Realität.

In dieser Arbeit wird der Real2Sim-Gap verursacht durch die Verschiebungen des Objektes betrachtet. Greifer, montage…

选用的抓手是啥，真空？ 就叫 SPT\_Vacuumtool？

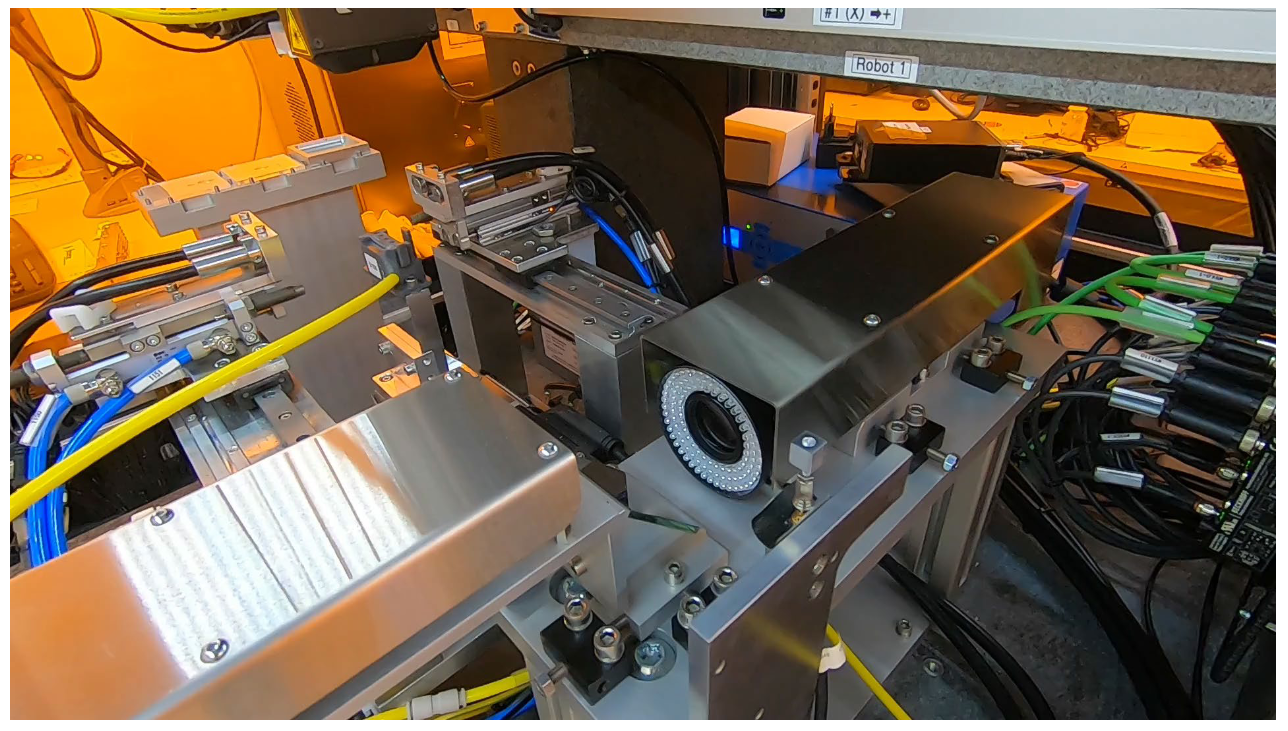


Arbeitsvorbereitung

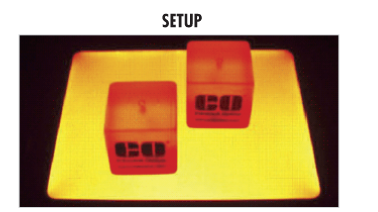
我们要检测的是内圈的圆

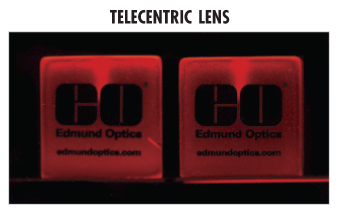
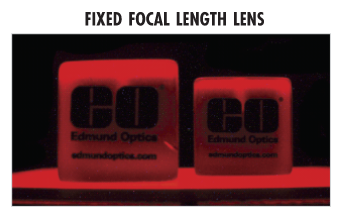
Die Abbildung 1 zeigt die verwendet Präzisionsmontageanlage in dieser Arbeit. Davon relevant für die Kalibrieraufgabe dieser Arbeit sind die Unterseitenkameta, der Spiegel sowie die vier Bewegungsachsen. Der zu kalibrierte Bauteil ist der Vakuumgreifer ‚SPT\_Vacuumtool‘, der 真空吸盘. Die verwendete Programmiersprache ist Python(Version: 3.10.6). Die Programmierumgebung ist Visual Studio Code. Das Betriebssystem ist Linux.

Das in der Unterseitenkamera verwendete Objektiv ist ein telezentrisches Objektiv. Eine Darstellung ist wie die Abbildung几 gezeigt.



Ein telezentrisches Objektiv ist häufig verwendet für den industriellen Zweck wegen seine hohe Genauigkeit. Telezentrische Objektive haben ein konstantes Bildfeld und keinen Blickwinkel, wodurch den Parallaxenfehler vermieden werden, die bei herkömmlichen Objektiven auftreten. Telezentrische Objektive haben bei jeder Objektentfernung das gleiche Bildfeld. Bei herkömmlichen Objektiven ändert sich der Bildwinkel mit der Entfernung zwischen Objektiv und Objekt. Mit zunehmender Entfernung nimmt die Vergrößerung ab. Dies ähnelt der menschlichen Wahrnehmung und entspricht unserer Tiefenwahrnehmung. Die Bildwinkel erzeugen Parallaxen- oder Perspektivenfehler. Dies ist der Beobachtungsfehler bei Messungen des Bildverarbeitungssystems aufgrund der offensichtlichen Verschiebung, die durch die beiden Objekte hintereinander verursacht wird, wenn zwei Objekte beobachtet werden. Die Genauigkeit der Bildverarbeitungssystemmessung wird dadurch verringert.





Ein Beispiel ist wie das Bild.. gezeigt. Daraus ist es deutlich zu sehen, wenn ein unterschiedlicher Abstand der beiden Objekte zu der Kamera liegen, wird das durch die konventionellen Objektiven aufgenommene Bildgröße ganz unterschiedlich. Wird ein telezentrisches Objektiv verwendet, tritt dieses Problem nicht auf.

Situationen, die in diesem Arbeit auftreten können, ist das wie in der Abbildung几 gezeigte Ablesen eines Zeigermessgeräts. Wenn es nicht im rechten Winkel auf die Skala blickt wird, führt es zu einer Messabweichung beim Ablesen. Da zwischen Zeiger und Skale ein kleiner Abstand besteht, wird ein falsches Messergebnis durch die Parallaxe erhält. In unserem Fall, da es zu Beginn unmöglich ist, den Greifer senkrecht gegenüber der Kamera zu bewegen und eine spätere Rotaion des Greifers ausgeführt wird, weicht der Greifer von der Mitte der Kamera ab. Der resultierende Parallaxenfehler wird deswegen auftreten.

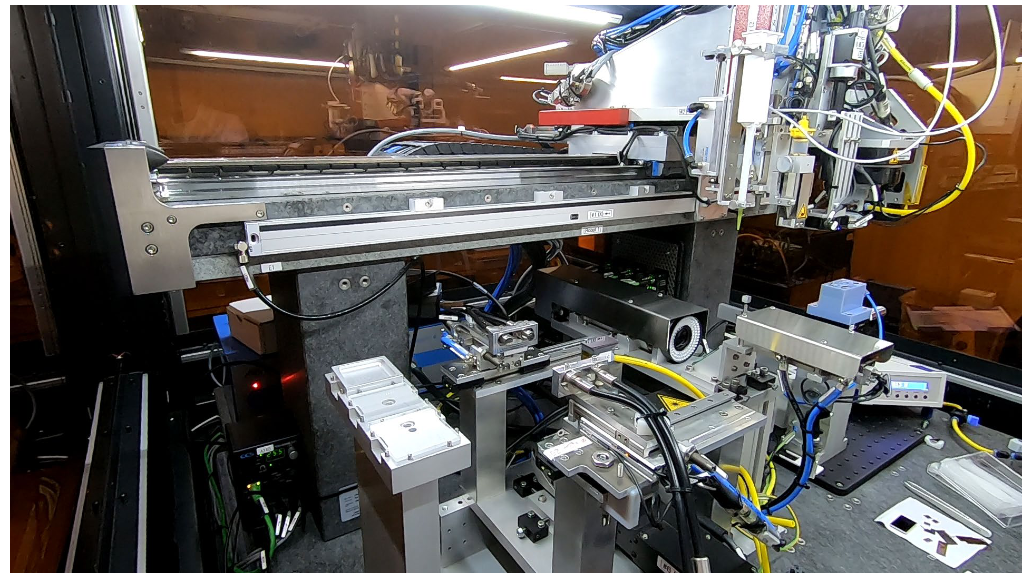
Wegen der Benutztung des telezentrischen Objektive brauchen diese Parallaxenfehler nicht zu berücksichtigen. Dies erleichtert auch die folgenden Berechnungen in dem Programm. Außerdem vermeidet das in dieser Aufagabe verwendete telezentrische Objektive das Problem der Verzeichnung, was die Genauigkeit der Vermessung weiter verbessert.



https://de.wikipedia.org/wiki/Parallaxenfehler#/media/Datei:Parallaxenfehler\_Spiegelskala.jpg

https://www.edmundoptics.de/knowledge-center/application-notes/imaging/advantages-of-telecentricity/

https://de.wikipedia.org/wiki/Parallaxenfehler



检测逻辑，先运动，然后**画圈**，然后检测。

Basierend auf dieser Gedanke wird die Aufgabe hauptsätzlich in zwei Teile eingeteilt wird. 。。。

2.2 Das Robot Operating System

Das Robot Operating System, abgekürzt mit ROS, ist ein Framework für persönliche Roboter und Industrieroboter. ROS ist kein Betriebssystem. Vielmehr handelt es sich um ein SDK (Software Development Kit), das die Bausteine bereitstellt, die zum Erstellen der Roboteranwendungen benötigt sind. Diese Arbeit wird mit ROS als den Rahmen durchgeführt.



Abbildung 1: Das ROS-Ökosystem[[2]](#footnote-3)

Wie die Abbildung 1 gezeigt, besteht das ROS-Ökosystem aus vier Teilen: ‚plumbing‘, ‚tools‘, ‚capabilities‘ und ‚community‘. Im Kern stellt ROS ein Nachrichtenübermittlungssystem bereit, das oft als „Middleware“ oder „Plumbing“ bezeichnet wird. Dies Nachrichtensystem verwaltet die Details der Kommunikation zwischen ‚Node‘. Ein Node ist ein Prozess, der Berechnungen durchführt. Nodes werden zu einem Diagramm zusammengefasst und kommunizieren über ein anonymes Publish/Subscribe-Modell. Bei der Entwicklung von Roboteranwendungen gibt es viele Schwierigkeiten und die ‚tools‘ in ROS helfen gut dabei. ROS hat viele Werkzeuge einschließlich: Start, Selbstbeobachtung, Debugging, Visualisierung, Plotten und so weier. Außerdem bietet ROS die Bausteine von Treibern über Algorithmen bis hin zu Benutzeroberflächen, was die ‚capabilities‘ von ROS verkörpert. Die ‚community‘ von ROS ist groß, vielfältig und global. Ingenieure und Hobbyisten aus den ganzen Welt halten das ROS-Projekt am Laufen.

https://www.ros.org/blog/ecosystem/

http://wiki.ros.org/Nodes

In dieser Arbeit wird die zweite Generation des Roboter-Betriebssystems(ROS 2) benutzt. ROS 1 wurde vom Robotik-Inkubator Willow Garage popularisiert. ROS 1 hat in fast allen Bereichen intelligenter Maschinen Wirkung gezeigt. Als die kommerzielle Möglichkeiten in die Produkte übergegangen, zeigten die Grundlagen von der ersten Generation des Roboter-Betriebssystems(ROS 1) als Forschungsplattform ihre Grenzen. In ROS 1 haben die Sicherheit, Netzwerktopologie und Systemverfügbarkeit keine Priorität. Aber mit der Entwicklung der Roboterindustrie werden die Sicherheit, Zuverlässigkeit in nicht-traditionellen Umgebungen und die Unterstützung großer eingebetteter Systeme von immer größer Bedeutung. Durch eine komplett neue Gestaltung, meistern ROS 2 diese Herausforderungen gut. ROS 2 basiert auf dem Data Distribution Service (DDS). Dieser ist ein offener Kommunikationsstandard, der in kritischen Infrastrukturen wie Militär, Raumfahrzeugen und Finanzsystemen verwendet wird. Es löst viele Probleme beim Aufbau zuverlässiger Robotersysteme. DDS ermöglicht ROS 2 erstklassige Sicherheit, eingebettete und Echtzeitunterstützung, Multi-Roboter-Kommunikation und Betrieb in nicht idealen Netzwerkumgebungen.

https://arxiv.org/pdf/2211.07752.pdf

kommunikation

Um das Steuern des Roboters mit ROS zu erreichen, muss die grundlegende Kommunikation zwischen ‚Nodes‘ ausgeführt. Ein ‚Node‘, das vorger schon erwähnt wurde, ist für einen einzigen, modularen Zweck in ROS verantwortlich. Ein vollständiges Robotersystem besteht aus vielen Nodes, die zusammenarbeiten. Jedes Node kann die Daten senden oder von anderen Noden die Daten empfangen, wodurch eine Kommunikation zwischen ‚Nodes‘ erreicht wird.

<https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>

ROS-Anwendungen kommunizieren typischerweise über unterschiedliche Typen der Kommunikationsmodi in ROS. In ROS 2 gibt es insgesamt drei Typen der Kommunikationsmodi zwischen Nodes: Topic, Service und Action. Die Abbildung 几 ist eine Darstellung der drei Modi in ROS 2. Diese Kommunikationsmodi basieren auf drei unterstützten Basisschnittstellen.

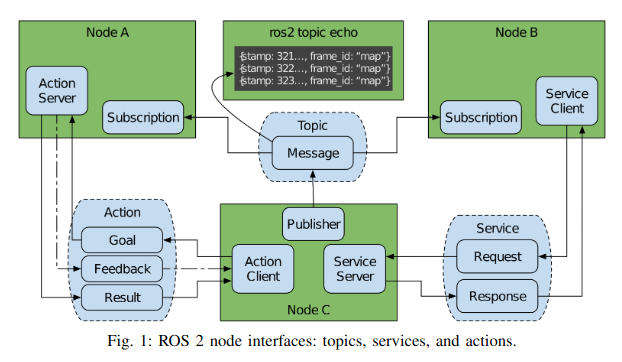
ROS 2 verwendet eine vereinfachte Beschreibungssprache, die Interface Definition Language (IDL), um diese Schnittstellen zu beschreiben. Diese Beschreibung erleichtert die ROS-Werkzeuge, den Quellcode für den Schnittstellentyp in mehreren Zielsprachen automatisch zu generieren. Die drei Schnittstellen sind: ‘msg’, ‘service’, ‘action’, welche bedeuten Message, Service und Aktion.

‘msg’: Eine Message in ROS ist verwendet in der Topic-Kommunikation und ist beschreibt durch die ‘.msg’-Datei, die aus einfache Texte besteht. Die ‘.msg’-Datei wird verwendet, um den Quellcode für Message in verschiedenen Programmiersprachen zu generieren.

‘service’: Ein Service entspricht der Service-Komunnikation und ist beschreibt durch ‘.srv’-Datei, die aus einer Anfrage und einer Antwort besteht. Die Anfrage und die Antwort sind jeweils eine Message.

‘action’: Eine Aktion entsprichy der Action-Kommunikation. Sie werden durch ‘.action’-Dateien beschreiben und bestehen aus drei Teilen: einem Ziel, einem Ergebnis und Feedback. Jeder Teil ist selbst eine Message.

https://docs.ros.org/en/foxy/Concepts/About-ROS-Interfaces.html

https://arxiv.org/pdf/2211.07752.pdf

(wie in Abbildung … topic)“Topic” ist eine Hauptmethode, mit denen die Daten zwischen den Nodes übertragen werden. Sie verwendet ein “Publisher-Subscriber”-Modell um einen Datenfluss zu erreichen. Die Nodes mit gleichen Namen der “Topics” kommunizieren miteinander. Ein Node kann die Daten zu einer beliebigen Anzahl von “Topics” publizieren und auch eine beliebige Anzahl von “Topics” abonnieren. “Service” basieren auf einem “Call-and-Response”-Modell. Anders als einen kontinuierlichen Datenfluss von der “Topic”-Kommunikation, stellt die “Service”-Kommunikation die Daten nur dann bereit, wenn sie ausdrücklich von einem Client aufgerufen werden. Es kann viele Clients existieren , die einen selben Service benutzten, aber für jeder Service kann es nur ein einzelner Server existieren. “Action” ist für lang laufende Aufgaben gedacht. Sie bestehen aus drei Teilen: einem Ziel, Feedbacks und einem Ergebnis. Die “Action”-Kommunikation benutzt ein “Client-Server”-Modell, das auf “Topic” und “Service” basiert. Ein „Action”-Client sendet ein Ziel an einen

, der das Ziel bestätigt und einen Feedbackstrom und ein Ergebnis zurückgibt. Eine Haupteigenschaft der “Action”-Kommunikation ist, dass sie abgebrochen während der Ausführung werden kann. Außerdem kann die “Action”-Kommunikation stetige Feedbacks während der Ausführung neben der einzelnen endlichen Antwort des Ergebnisses geben.

Neben dieser Haupt-Kommunikationsmethoden gibt es noch ‘Parameter’

3. Hauptteil(整体结构):

设备的结构，例如拍摄是通过镜子折射进行的。

Der erste Schritt ist die Positionierung des Greifers bzw. die richtige Bewegung des Greifers über den Spiegel.

远光镜

3.3 Trajektorie

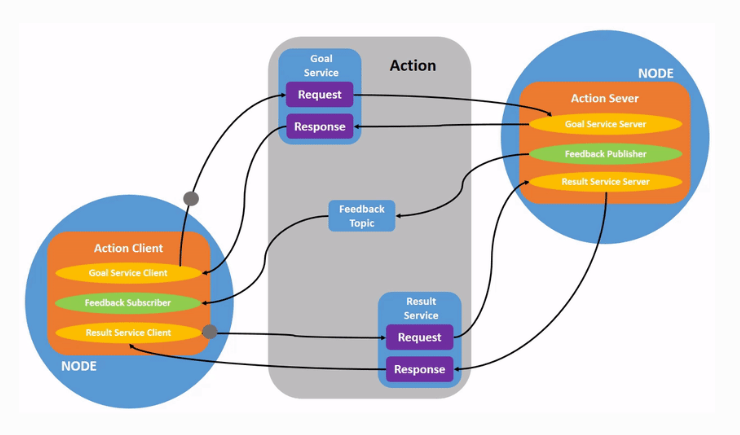
Es wird in dieser Arbeit (Trajektorie) verwendet, um den Roboter zu steuern. 和forwaord position对比的优劣？

Um die Trajektorien an den Controller zu senden kann man zwei Mechanismen davon benutzten: “Action Interface” oder “Topic Interface”. “Topics” werden normalerweise für kontinuierliche Datenströme wie Sensordaten, Roboterstatus usw. verwendet, wobei der Datenfluss unidirektional ist. “Actions” werden meistens für diskretes Verhalten verwendet, das zur Steuerung eines Roboters oder zum Laufen über einen längeren Zeitraum. Während der Ausführung der “Actions” werden die Feedbacks gegeben. Im unseren Fall wird ein Präzisionsmontageroboter gesteuert und die Feedbacks während dem Prozess sind erwünscht, deswegen wird in dieser Arbeit die Steurung des Roboter durch die Kommunikation mit “Action” erreicht.

Für die Aufgabe dieser Arbeit ist das Steuern durch die ‘Action’-Kommunikation besser geeignet wegen der Möglichkeit des Bebuggings während der Bewegung. Deshalb wird das Pragramm zur Steuerung des Greifers in dieser Arbeit rund um diesen Kommunikationsmodus erweitert.

Unten ist eine grafische Darstellung der “Action”-Kommunikation.

Der “Action Client” fordert ein Request an den “Action Server” an und der “Action Server” sendet eine Antwort zurück, um es aufzuweisen, ob das Request akzeptiert wird. Wenn das Request erfolgreich akzeptiert wird, wird das Request bzw. die “Action” ausgeführt. Im Anschluss an die “Request”-Anforderung stellt der Client eine asynchrone Anforderung für das Ergebnis. Während das Ziel ausführt wird, gibt der “Action Server” die kontinuierlichen Feedbacks an den “Action Client”. Und letztendlich gibt der “Action Server” ein einmaliges Feedback, um zu deuten ob die “Action” erfolgreich beendet wird. 然后也许可以结合一下我们的程序，带进去讲一下



控制greifer运行到一个位置，是有一定偏差的，很微小，可以到时候看看是小数点后几位，可能影响到最后的精度 allow\_partial\_joints\_goal?

讲trajectory代码的时候继续用这个链接：

http://wiki.ros.org/joint\_trajectory\_controller

3.

3.1 Die Steurung des Roboters

Genauso wie die Funktionsweise der “Action”-Kommunikation wird das Programm des Steuerns von dem Greifer implemeiert. Die Funktion für die Bewegung der Positionierung ist align\_action.

Zuerst wird die Trajektorie definiert. Um einen unmittelbaren Hinweis zu geben, mit dem ein Startsignal der Funktion mitgeteilt wird, wird eine Info “Starting align…” gegeben. Anschließend folgt die Definition der Bewegung. Die Trajektorie ist angegeben durch die Message ‘trajectory\_msgs/msg/JointTrajectoryPoint’, die wie folgend definiert:

double[] positions

double[] velocities

double[] accelerations

double[] effort

builtin\_interfaces/msg/Duration time\_from\_start

Die erste vier Elemente verwenden den Datentyp ‚double‘. ‚double‘ ist ein DDS-Typ, der entspricht dem ROS-Typ ‚float 64‘ und dem Python-Typ ‚float‘. Der Datentyp des letzten Elementes kann durch den ROS-Typ ‚int32‘ oder ‚uint32‘ oder Python-Typ ‚int‘ angegeben. ‚position‘ gibt an, wo sich die Zielposition befindet. Die Zielposition ist so gewählt, dass das ‚SPT\_Vacuumtool‘ ungefähr in der Mitte des Kamerasichtfeldes steht, womit eine spätere Rotation erfolgreich ausgeführt werden kann. Außerdem soll die Höhe des ‚SPT\_Vacuumtool‘ richtig gewählt werden, sodass es in der Bildverarbeitung besser behandelt werden kann. Bei der realen Anlage wird die Zielposition auf [-0.359, -0.0458, 0.03, -1200000.0] eingestellt und in der Simulationsumgebung wird sie auf [-0.359, -0.0458, -0.051544, 0.0] eingestellt. Beide Werte werden getestet und sind in der entsprechenden Situation sinnvoll. ‚velocities‘ und ‚accelerations‘ stellen es fest, welche Werte die Geschwindigkeit und die Beschleunigung sind wenn das Robot an dem Zielpunkt ankommt. ‘effort’ beschreibt das Drehmoment oder die Kraft, die an jedem Gelenk ausgeübt werden. In unserem Fall ist dieser Wert nicht relevant. Das letzte Element ‚Duration‘ gibt die gewünschte Zeit vom Start der Trajektorie bis zum Erreichen des Zielnpunkts. Für eine stabile Bewegung, wird dieser Wert auf 6 Sekunden eingestellt. Dieser Wert ist nur in der Simulationsumgebung relevant. Um eine Trajektorie richtig einzustellen, muss es nur einer von ‚velocities‘ und ‚Duration‘ ausgewählt wird. Andernfalls wird die Bewegung des Greifers fehlerhaft wegen eines Widerspruchs dieser zwei Werte zueinander.

https://docs.ros.org/en/foxy/Concepts/About-ROS-Interfaces.html

https://docs.ros2.org/foxy/api/trajectory\_msgs/msg/JointTrajectoryPoint.html

Nachdem die Trajektorie definiert wird, muss sie dann an den Roboter gesendet werden.

Wie oben erwähnt, wird in dieser Arbeit die ‚action‘-Kommunikation benutzt zum senden der Trajektorie. Nach Eingabe des Befehls ‚ros2 action list -t‘ in dem Terminal werden die vorhandenen Aktionen inklusive der Namen ihrer Schnittstellen angezeigt. Es ist dann offensichtlich eine ‚Action‘ heißt /joint\_trajectory\_controller/follow\_joint\_trajectory existiert. Ihre Schnittstelle ist control\_msgs/action/FollowJointTrajectory. In dieser Schnittstelle sind folgende vier Elemente relevant, um den Roboter zu steuern:

trajectory\_msgs/JointTrajectory trajectory

JointTolerance[] path\_tolerance

JointTolerance[] goal\_tolerance

duration goal\_time\_tolerance

http://docs.ros.org/en/noetic/api/control\_msgs/html/action/FollowJointTrajectory.html

Davon wird nur das Element trajectory\_msgs/JointTrajectory trajectory benutzt. Die andere sind die unterschiedlichen Toleranz, um die Trajektorie weiter zu begrenzen. Werden sie nicht manuell angegeben, werden die Werte auf die Standardwerte für den „Action”-Server gesetzt. Dies Element ist wie folgend definiert:

Header

string[] joint\_names

JointTrajectoryPoint[] points

Es besteht aus drei Elemente: header, joint\_names und points, wobei points vorher schon definiert wird. Das Element header hat zwei Felder: stamp und frame\_id. Diese zwei Elemente sind nicht relevant für die Definition einer Trajektorie und werden deswegen nicht weiter diskutiert. joint\_names gibt an, welche Gelenke zu steuern sind. Im ersten Schritt müssen vier Gelenke gesteuert werden: 'X\_Axis\_Joint', 'Y\_Axis\_Joint', 'Z\_Axis\_Joint', 'T\_Axis\_Joint'. Diese Namen müssen den, die in der Datei ‚pm\_robot.xacro‘ dediniert werden entsprechen. ‚pm\_robot.xacro‘ ist eine Datei, die die Konstruktion des Roboters beschreibt. Um die einzelne Rotation zu erreichen, die in dem folgenden Teil dieser Arbeit nötig ist, muss dazu das Parameter allow\_partial\_joints\_goal hinzugefügt und dessen Wert auf ‚True‘ eingestellt werden. Sonst müssen alle vier Gelenke definiertet werden, was zu einer Bewegung dieser alle vier Gelenke und einer misslungenen Detektion führt.

Schließlich werden die Feedbacks der Bewegung definiert, womit eine vollständige Bewegung der Positionierung beendet.

Das ganze Programm wird wie folgend durchführt. Zuerst wird die Methode align\_action() definiert, wobei die Class ‚AutoCalibration‘ übergeben. Das Code ist wie unten gezeigt. Es wird die Methode get\_logger() des Nodes und dessen Methode info() Verwendet um den Startsignal von align\_action() zu senden. Danach wird die Class JointTrajectoryPoint() instanziiert. 。。传参，duration。 Dann wird die Class FollowJointTrajectory.Goal() instanziiert um die Trajektorie zu seden. Darauf folgt die Definition der Attribute joint\_names und points der Class. Um das Ziel erfolgreich zu senden, wird die Methode wait\_for\_server() der Class ActionClient() verwendet. Diese Methode wartet darauf, dass der Aktionsserver verfügbar ist, und sendet dann ein Ziel an den Server. Dann wird die Methode send\_goal\_async() der Class ActionClient() verwendet. Diese Methode sendet ein Ziel und erhält asynchron das Ergebnis, das als ein ‚Future‘ zurückgegeben. Das Ergebnis wird auf ein ‚ClientGoalHandle‘ gesetzt, wenn der Empfang des Ziels von einem Aktionsserver bestätigt wird. Der Hauptteil des Programms, das die Trajektorie sendet, ist hier fertig. Um zuletzt die Rückmeldung zu bekommen, ist es benötigt, die ‚callback‘-Funktion zu addieren. Das erfolgt durch die Methode add\_done\_callback() der ‚Future‘. Mit dieser Methode kann eine ‚callback‘-Funktion eingestellt, die sofort aufgerufen werden kann, wenn die ‚Future‘ erledigt ist. Ausserdem wird eine ‚callback‘-Funktion benötigt, um die Rückmeldungen während den Prozess zu bekommen.

def align\_action(self):

self.get\_logger().info('Starting align...')

target\_point = JointTrajectoryPoint()

target\_point.positions = Parameter[3]

target\_point.time\_from\_start = Duration(sec= 6)

goal\_msg = FollowJointTrajectory.Goal()

goal\_msg.trajectory.joint\_names = ['X\_Axis\_Joint','Y\_Axis\_Joint',

'Z\_Axis\_Joint','T\_Axis\_Joint' ]

goal\_msg.trajectory.points = [target\_point]

self.action\_client.wait\_for\_server()

self.send\_goal\_future = self.action\_client.send\_goal\_async(goal\_msg

,feedback\_callback=self.align\_callback)

self.send\_goal\_future.add\_done\_callback(self.align\_response\_callback)

Anschließend erfolgt die Definitionen der ‚callback‘-Funktionen.

def align\_response\_callback(self, future):

goal\_handle = future.result()

if not goal\_handle.accepted:

self.get\_logger().info('Goal rejected.')

print(future.exception())

return

self.get\_logger().info('Goal accepted.')

self.get\_result\_future = goal\_handle.get\_result\_async()

self.get\_result\_future.add\_done\_callback(self.align\_result\_callback)

def align\_result\_callback(self,future):

# error\_code = future.result().result.error\_code

# if error\_code != 0:

# self.get\_logger().info(f'Error code: "{error\_code}"')

self.get\_logger().info('Goal reached!')

def align\_callback(self,feedcak\_msg):

self.get\_logger().info('Approaching...')

说程序里定义了几秒

Um möglichst viele Punkte zu sammeln bzw. eine möglichst bessere Interpolation des Kreises zu machen und auch gleichzeitig die maximale Rotationsgrenze des Greifers zu berücksichtigen, wird die Rotation 角度这样定义。。Eine Drehung in einem größeren Winkel kann das mit dem Greifer verbundene Kabel beschädigen，同时还要考虑ursprüngliche Position.

Yaml

坐标转换用youtube里的图

5. OpenCV

主要思想

然后几个重要的函数

然后几个不重要的函数，比如画圆啥的

OpenCV ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und Computer Vision. Sie besitzt mehrere Hundert Computer Vision-Algorithmen. Die Entwicklung der Bibliothek wurde von Intel initiiert und bis 2013 von Willow Garage gepflegt und nachher wieder von Intel übernommen wurde. OpenCV besitzt eine schnelle Geschwindigkeit und eine große Menge der Algorithmen aus neuesten Forschungsergebnissen. Außerdem werden die Algorithmen in die entsprechenden Funktionen gekapselt und deswegen ist es ziemlich bequem um die bestimmten Aufgaben zu erledigen mit OpenCV.

https://de.wikipedia.org/wiki/OpenCV

<https://docs.opencv.org/4.7.0/d1/dfb/intro.html>

In dieser Arbeit wird OpenCV hauptsätzlich benutzt zur Detektion des Vakuumgreifers. Dafür wird es zuerst die Verarbeitung zu den aufgenommenen Bilder implementiert. Wegen einer kreisförmigen Kontur des Greifers besteht der zweite Schritt darin, den Kreis zu detektieren. Dazu werden in dieser Arbeit diese folgenen Hauptfunktionen verwedent: findContours(), Canny(), fitEllipse() und GaussianBlur().

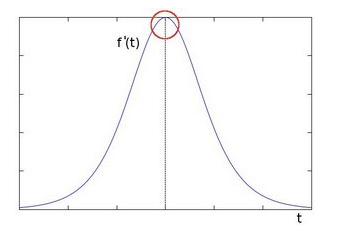
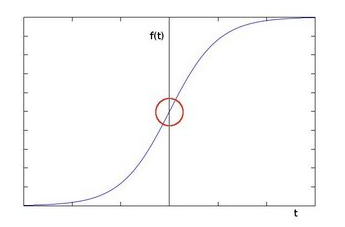
Eine Bildvorhandlung wird immer vor der Detektion implementiert. Aber für eine bessere Erklärung und ein besseres verständnis wird der Gaußsche Filter im letzten Kapitel erläutert.

2. Canny Funktion

Bei Bildern sind Kanten durch plötzliche Änderungen der Pixelintensität gekennzeichnet. Eine grafische Darstellung ist wie z. B. im Bild 几 gezeigt. Es ist offensichtlich, dass eine Änderung der Pixelintensität zwischen dem Gesicht und den Haare.

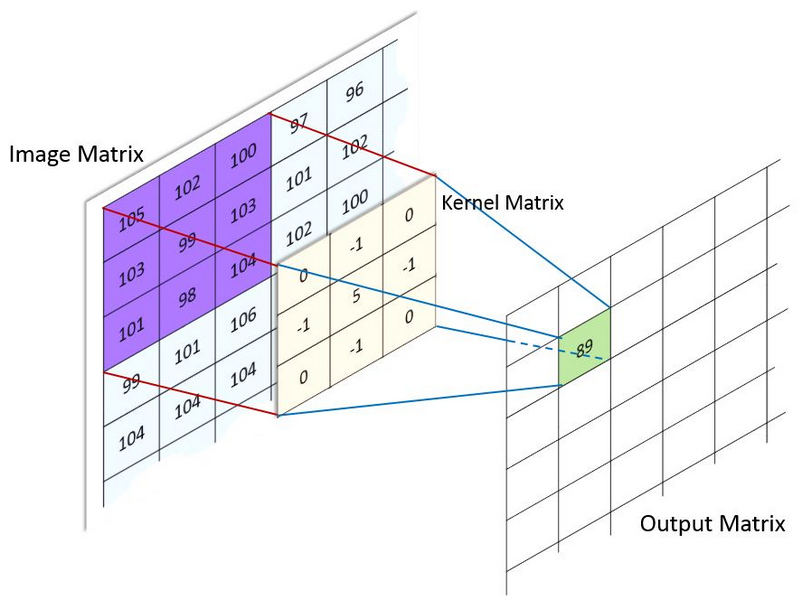


Eine Intensitätsänderung ist beispielsweise in der Abbildung 1 zu sehen. Die Abbildung 1 zeigt den Pixelintensitätstrend eines 1D-Bildes, wobei die x-Achse die Zeit und y-Achse die Pixelintensität beschreibt. Eine Kante wird durch den ‚Sprung‘ der Intensität wie in dem Bild angezeigt. Wird die erste Ableitung genommen, ist der ‚sprung‘ besser zu erkennen. Eine Darstellung ist wie das Bild 几 gezeigt. Die Ableitung beschreibt die Änderungsgeschwindigkeit an einem Punkt, mit dem kann die Änderung der Pixelintensität in einem Bild gut repräsentiert. Somit kann eine Methode zum Erkennen von Kanten in einem Bild durchgeführt werden, indem die Pixelpositionen lokalisiert werden, deren Gradienten höher als ihre Nachbarn sind.



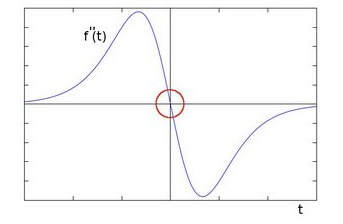
In OpenCV gibt es mehrere Möglichkeiten um die Kanten zu detektieren z. B. Sobel-Operator, Laplace-Operator und Canny-Operator. Davon nutzt Sobel-Operator diese Idee. Der Sobel-Operator ist ein diskreter Differenzierungsoperator. Es berechnet eine Näherung des Gradienten einer Bildintensitätsfunktion in beiden x- und y-Richtung, was durch die Faltung mit einem Kernmatrix berechnet. Die größeren Kerne ergeben eine bessere Annäherung an die Ableitung. Die kleineren Kerne reagieren empfindlicher auf Rauschen.

In einem 2D-Bild wird die Faltung in einer Form von Matrix implementiert, siehe Abbildung 几. Die Werte an den entsprechenden Positionen der Matrix werden multipliziert und schließlich addiert. Die bekommende Summe ist der Wert des entsprechenden Pixels.

https://github.com/meredithheskethfortescue/MIREVI\_FMA\_IntSys\_SS19

Ein Sobel-Operator ist einfach zu implementieren und schnell zu berechnen. Aber er reagiert empfindlich auf diagonale Kanten und erzeugt dicke Kanten, was die Genauigkeit einer weiteren Detektion beeinflusst.

Die Grundidee in Laplace-Operator um die Kanten zu detektieren ist ähnlich wie den Sobel-Operator. Der Laplace-Operator benutzt die zweite Ableitung der Intensitätsänderung eines Bildes. Wie vorher erwähnt, die Intensitätsänderung einer Kante wird dargestellt als einen ‚Sprung‘ der ersten Ableitung. Das bedeutet, wird die erste Ableitung genommen, ist dieser ‚Sprung‘ ein lokales Maximum. Wenn jetzt die zweite Ableitung genommen wird, ist es nicht schwer zu finden, dass der Wert des punktes, an dem ein Maximum sich befindet, in der zweiten Ableitung Null ist(siehe Abbildung 几). In Laplace-Operator wird diese Eigenschft der Ableitung verwendet, um die Kanten zu detektieren. Der Vorteil des Laplace-Operators besteht darin, dass er isotrop ist, d. h. er erkennt Kanten in allen Richtungen gleichermaßen und erzeugt dünne, scharfe Kanten. Der Nachteil ist, dass es sehr rauschempfindlich ist, da es kleine Änderungen der Pixelintensität verstärkt und außerdem keine Informationen über die Kantenausrichtung oder Ausrichtung liefert.



https://www.linkedin.com/advice/1/what-advantages-disadvantages-using-sobel-canny

https://docs.opencv.org/4.x/d2/d2c/tutorial\_sobel\_derivatives.html

https://learnopencv.com/edge-detection-using-opencv/

Canny-Operator bietet verleichsweise eine bessere Leistung. Canny Edge Detection ist ein beliebter Kantenerkennungsalgorithmus. Es wurde von John F. Canny entwickelt.

Die Canny() Funktion findet die Kanten in den Eingabebildern mit dem Canny Algorithmus. Canny-Edge-Detection-Algorithmus ist ein mehrstufiger Algorithmus, der die folgenden Stufen enthält(下面名字都用德语吧): **Noise Reduction, Finding Intensity Gradient of the Image, Non-maximum Suppression und Hysteresis Thresholding.**

由于canny的优点：不容易受噪声干扰 --→ 不用用滤波也许

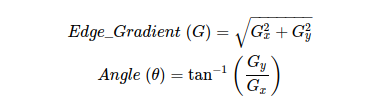
**Noise Reduction:**

Da die Kantenerkennung anfällig für Bildrauschen ist, besteht die erste Stufe darin, das Bildrauschen zu entfernen. Dies wird im nächsten Kapitel genauer besprochen.

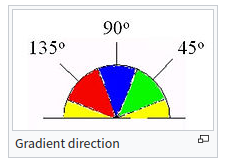
也许可以先讲讲canny的参数啥的

**Finding Intensity Gradient of the Image:**

Der Gradient eines Bildes ist einer der Grundbestandteile der Bildverarbeitung und Computer Vision. Er repräsentiert eine Richtungsänderung der Intensität oder Farbe in einem Bild. In diesem Schritt wird das vorher geglättete Bild mit einem Sobel-Kern gefiltert in beide horizontaler und vertikaler Richtung. Die erste Ableitung in der horizontalen und vertikalen Richtung ist damit ausgerechnet. Der Gradient und die Richtung der Kante für jedes Pixel sind wie das folgende (Bild) gezeigt:



Die Richtung des Gradienten ist immer senkrecht zu den Kanten und sie wird auf einen von vier Winkeln gerundet: die vertikale, horizontale und zwei diagonale Richtungen. Eine grafische Darstellung ist wie folgend gezeigt. Die Winkel in [0°, 22,5°] oder [157,5°, 180°] wird auf 0° abgebildet.



https://en.wikipedia.org/wiki/Image\_gradient

theata 正负详解：

https://pyimagesearch.com/2021/05/12/image-gradients-with-opencv-sobel-and-scharr/

**Non-maximum Suppression: 重要：比sobel得到的更光滑：（ppt里用）https://medium.com/@haidarlina4/sobel-vs-canny-edge-detection-techniques-step-by-step-implementation-11ae6103a56a**

为了细化边缘才做的这一步

Wie der Name dieses Schritts werden im diesen Schritt die schärfsten Gradienten beibehalten und die anderen werden verworfen, womit eine Kantenverdünnung erreicht wird. Nachdem die Größe und Richtung des Gradienten ermittelt wurde, wird ein vollständiger Scan des Bildes durchgeführt, um alle unerwünschten Pixel zu entfernen, die möglicherweise nicht die Kante bilden. Dazu wird bei jedem Pixel geprüft, ob es in seiner Nachbarschaft in Gradientenrichtung ein lokales Maximum gibt, siehe Abbildung 几.



Der Punkt A liegt auf einer Kante. Die Richtung des Gradienten ist vertikal zu der Kante. B und C sind zwei Punkte, die auch auf der Richtung des Gradienten liegen. In diesem Fall wird der Punkt A mit Punkt B und C überprüft, um zu sehen, ob er ein lokales Maximum bildet. Wenn ja, bleibt der Punkt A erhalten in die nächste Stufe, andernfalls wird es unterdrückt (auf Null gesetzt).

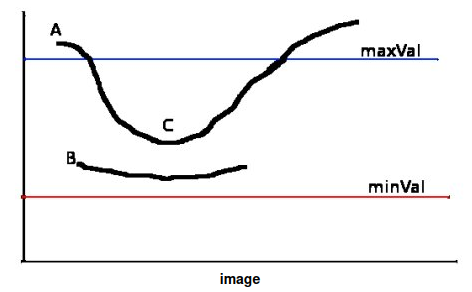
Deswegen 正如前面参数部分所说的输出是一个而值化的图。In short, the result you get is a binary image with "thin edges".

https://en.wikipedia.org/wiki/Canny\_edge\_detector

Hysteresis Thresholding:

Auf dieser Stufe wird entschieden, welche Kanten wirklich Kanten sind und welche nicht. Das ‚Hysteresis’ hier bedeutet die Abhängigkeit des Zustands eines Systems von seiner Geschichte. In der Canny() Funktion kann es als die Abhängigkeit eines Pixels von seinen Nachbarn verstanden werden. Das wird im folgenden erklärt werden. Um zu unterscheiden, ob es sich um eine reale Kante handelt werden in Canny() zwei Schwellenwerte verwendet werden: minVal und maxVal, welche bedeuten der minimale Wert und der maximale Wert. Die Kante mit einem Intensitätsgradienten größer als maxVal werden als reale Kanten betrachtet und diejenigen unterhalb von minVal werden keine Kanten betrachtet und werden daher verworfen. Diejenigen, die zwischen diesen beiden Schwellenwerten liegen, werden nach ihrer Konnektivität als Kanten oder Nichtkanten klassifiziert. d.h. wenn sie mit „sicheren Kanten“-Pixeln verbunden sind, werden sie als Teil von Kanten betrachtet. Andernfalls werden sie ebenfalls verworfen.

Siehe das Bild unten:



Die Kante A liegt über dem maxVal und wird daher als reale Kanten betrachtet. Obwohl Kante C unter maxVal liegt, ist sie mit Kante A verbunden, deswegen wird sie auch als gültige Kante betrachtet. Ebenso ist der obere rechte Teil von Kante C, der über dem maxVal liegt. Daher erhalten wir die vollständige Kurve. Jetzt betrachten wir die Kante B. Obwohl sie über minVal liegt und sich im gleichen Bereich wie Kante C befindet, ist sie mit keinen realen Kanten verbunden und wird folglich verworfen. Daher ist es von wesentlicher Bedeutung, dass wir das geeignete minVal und maxVal auswählen, um das richtige Ergebnis zu erhalten. Außerdem werden in dieser Phase die kleine Pixelrauschen auch entfernt, da es davon ausgegangen wird, dass es sich bei den Kanten um lange Linien handelt. Und was wir endlich erhalten sind starke Kanten im Bild.

因此不需要去椒盐噪声了

Durch den Prozess des Canny-Edge-Detection-Algorithmuses ist es zu sehen, dass Canny-Algorithmus dünne und präzise Kanten erzeugen kann. Außerdem kann er sich durch einstellbare Parameter an unterschiedliche Bildeigenschaften anpasst werden. Die Nachteile bestehen darin, dass er relative komplexer und rechenintensiver ist und eine manuelle Abstimmung der Parameter erfordert, um optimale Ergebnisse zu erzielen.

https://docs.opencv.org/4.x/dd/d1a/group\_\_imgproc\_\_feature.html#ga04723e007ed888ddf11d9ba04e2232de

https://docs.opencv.org/4.x/da/d22/tutorial\_py\_canny.html

https://bbs.elecfans.com/jishu\_2314292\_1\_1.html

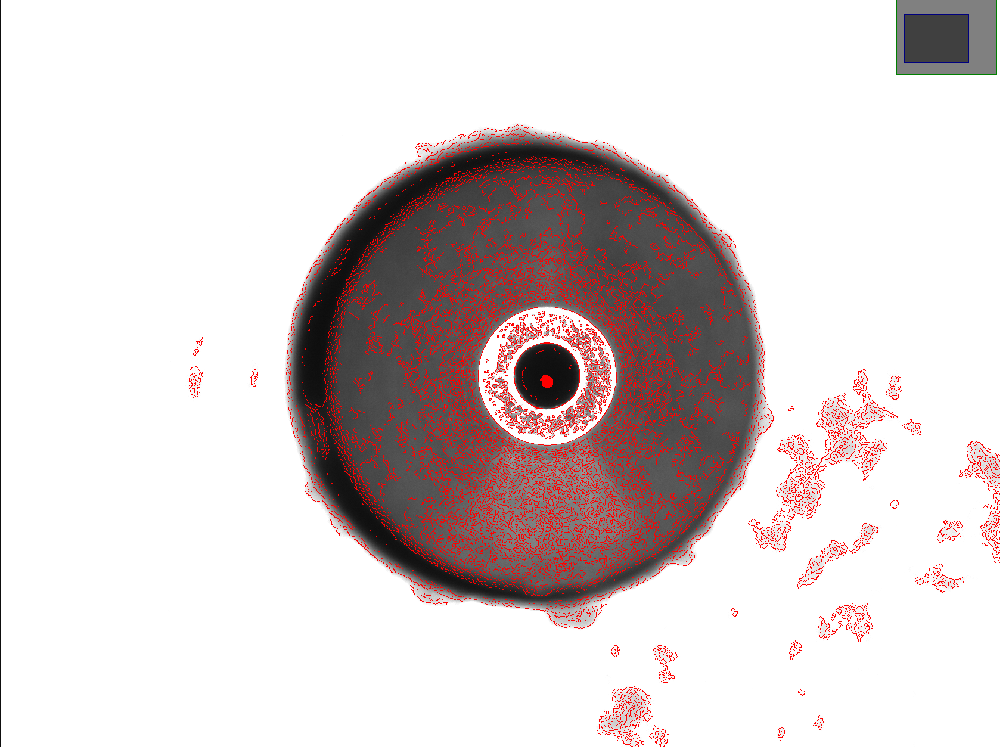
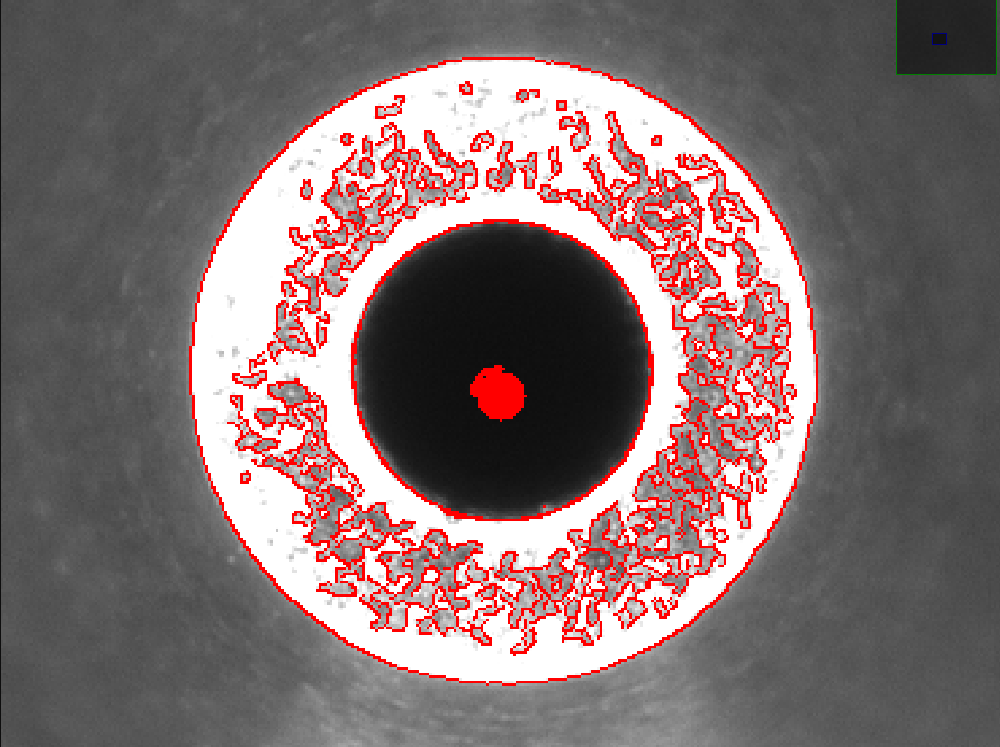
https://indiantechwarrior.com/canny-edge-detection-for-image-processing/

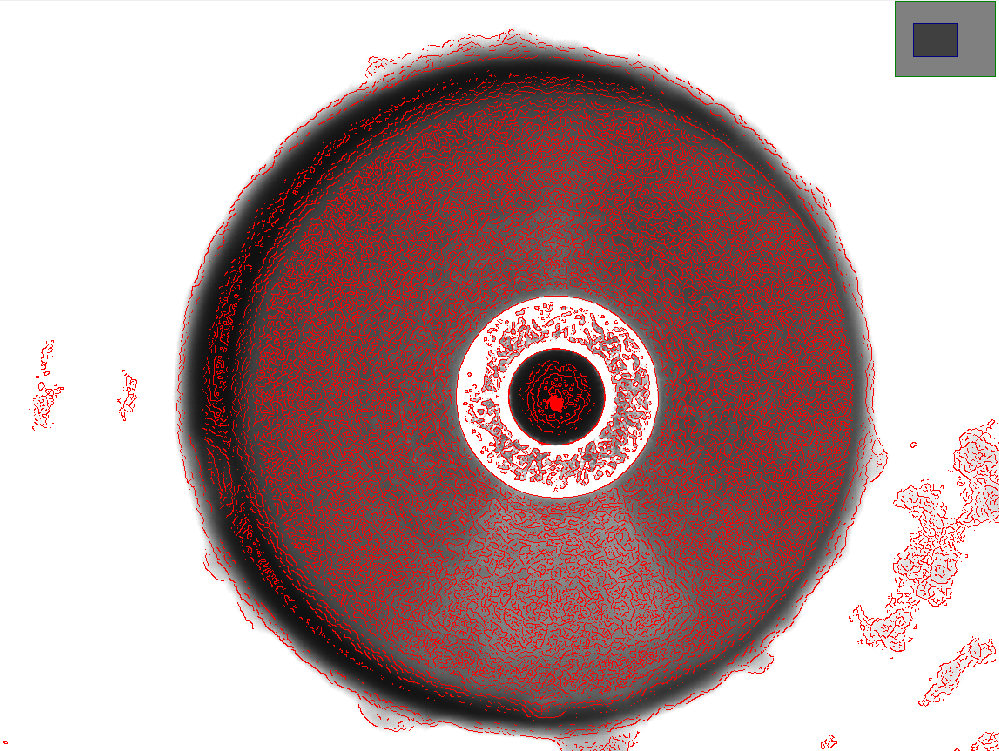
https://en.wikipedia.org/wiki/Hysteresis

In OpenCV, neben der Standradfunktion Canny(), gibt es noch eine „overloaded member function“. Eine „overloaded member function“ kann denselben Name wie die originale Funktion besitzen aber mit unterschiedlichen Argumenttypen oder unterschiedlicher Anzahl von Argumenten. Der Unterschied zwischen der Standradfunktion und der „overloaded funktion“ von Canny() liegt darin, dass die Eingabeparameter sind unterschiedlich. Die Standardfunktion verwendet ein Bild als die Eingabe, während die „overloaded member function“ die x- und y-Ableitung des Bildes verwendet. Im unseren Fall werden die Bilder direkt aufgenommen von der Kamera, deswegen wird die Standardfunktion von Canny() verwendet. Die Canny() Standardfunktion hat insgesamt 6 Parameter: „image“, „edges“, „threshold1“, „threshold2“, „apertureSize“ und „L2gradient“. „image“ ist das Eingabeparamter, das ein 8-bit-Bild anfordert. ‚edges‘ ist das Ausgabeparameter, das ein Einzelkanal-8-bit-Kantenbild ausgibt. ,L2gradient’ 。。。

Die anderen drei Parameter sind die Parameter des Canny-Edge-Detection-Algorithmus, die die Performance dieses Algorithmus beschreiben. ‚threshold1‘ und ‚threshold2‘ sind die zwei Schwellwerte für das Hystereseverfahren. Ihre Herangehensweise wird im obigen schon besprochen.

„apertureSize“ stellt die Größe der Kernmatrix für den Sobel-Operator fest. Unten sind die drei Bilder, die mit unterschiedlichen Parametern durchgeführt, dargestellt.





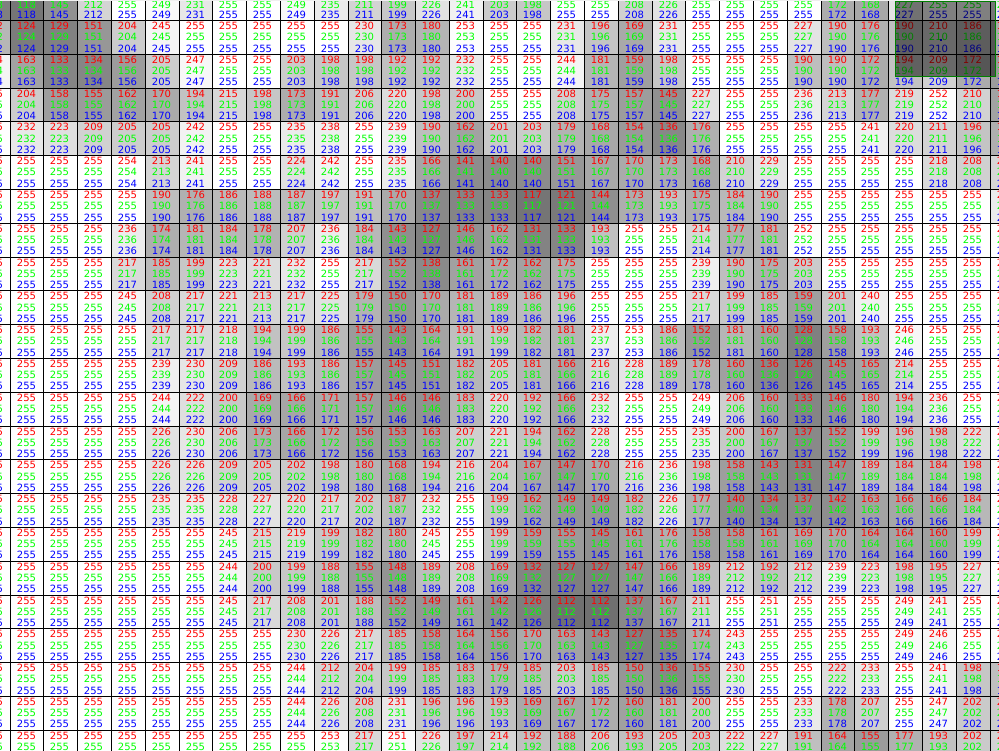
Der weiße Ring in den Bilder ist die untere Ebene des Vakuumgreifers. Aus diesen Bilder ist es ersichtlish, dass die Wirkung von der Kerngröße mit 5 und 7 unbefriedigend sind. Konturen werden an vielen unnötigen Stellen erkannt, wodurch mehr Berechnungen und ein viel größer Speicherraum kostet werden. Im Versuch mit Kerngröß 3 wird der Breich außer dem Ring fast keine Konturen detektiert, was eine bessere Wirkung in dieser Aufgabe zeigt.

2. gauss

Vor der Detektion eines Bildes wird es häufig zuerst eine Vorbehandlung impementiert. Damit kann Eine bessere Qualität der späteren Detektion erreicht werden.

Wird die Abbildung 1 betrachtet, ist es offensichtlich zu sehen, dass es viele schwarze Punkte auf der unteren Ebene des Vakuumgreifers bzw. auf dem weißen Ringbereich. Dies liegt an der Rauheit der unteren Ebene. Das Merkmal von diesem Rauschen ist ähnlich wie das Salz-und Pfefferrauschen. Das Salz-und Pfefferrauschen, auch Impulsrauschen genannt, ist eine Form von Rauschen. Dieses Rauschen kann durch scharfe und plötzliche Störungen im Bildsignal verursacht werden. Es zeigt sich als spärlich vorkommenden weißen und schwarzen Pixeln. Typischweise ist ein Medianfilter eine wirksame Methode zur Unterdrückung für diese Art von Rauschen.

Wird der Rauschenbereich vergrößert, ist beispielsweise



2. Da es schon mehrfach getestet wurde, wird es von diesem Kapitel an angenommen, dass der richtige Wert der Abweichung zwischen dem Vakuumgreifer und der Platte in der realität liegt zwischen ca. 18 µm und 19 µm und die Koordinate der Mitte des Greifers im Bildkoordinatensystem liegt ungefähr auf (1125 , 1040), sodass es in den unteren Kapitel die Beeinflussungen unterschiedlicher Parameter besser erklärt werden.

In der Bilderarbeitung werden die Konturen häufig verwendet zur Formanalyse sowie zur Objektdetektion und -erkennung. Konturen können einfach als eine Kurve erklärt werden, die alle angrenzenden Punkte (entlang der Grenze) derselben Farbe oder Intensität verbindet. Die Funktion findContours() wird verwendet, um die Konturen in einem Bild zu finden. In früheren Versionen von OpenCV modifiziert diese Funktion das Oringinalbild. Seit OpenCV 3.2 wird diese Charakteristik gelöscht. In OpenCV, das Finden von Konturen bedeutet das Finden weißer Objekte vor schwarzem Hintergrund. Deswegen soll das zu findende Objekt weiß und der Hintergrund schwarz sein.

Die Funktion findContours() gibt zwei Parameter zurück. Einer davon ist ‘contours’. Der Python-Typ von diesem Output ist ‘list’. Es enthält alle Konturen, die von dieser Funktion detektiert werden. Jede individuelle Kontur ist repräsentiert durch einem Numpy-Array von (x,y)-Koordinaten aller einzelnen Punkte des detektierten Objektes. Für eine bessere Genauigkeit der Detektion werden immer die Binärbilder gebraucht. Aber darüber muss man nicht viel nachdenken, weil das Output von Canny() Funktion ist schon...

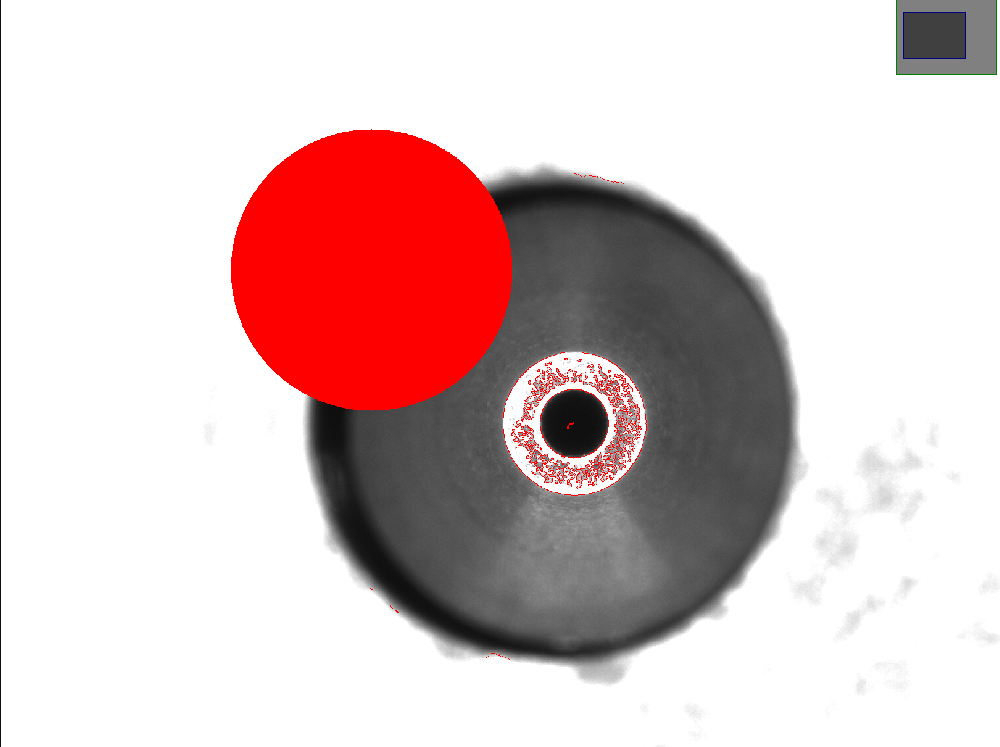
Ein anderes Output von findContours() ist ‚hierarchy‘. Wenn die Objekte in einem Bild detektieret werden, liegen in manchen Fällen einige Formen innerhalb anderer Formen. In diesem Fall bezeichnen wir das Äußere als Eltern und das Innere als Kind. Auf diese Weise stehen die Konturen in einem Bild in einer gewissen Beziehung zueinander. Und wir können angeben, wie eine Kontur miteinander verbunden ist, z. B. ob sie einer anderen Kontur untergeordnet ist oder ob sie ein übergeordnetes Element ist usw. Die Darstellung dieser Beziehung wird als Hierarchie bezeichnet. In OpenCV ist diese Beziehung zwischen Konturen als ein Array aus vier Werten repräsentiert: [Next, Previous, First\_Child, Parent]. ‚Next‘ bezeichnet den Index der nächsten Kontur auf derselben Hierarchieebene. ‚Previous‘ bezeichnet den Index der vorherigen Kontur auf derselben Hierarchieebene. ‚First\_Child‘ bezeichnet den Index seiner ersten untergeordnete Kontur und ‚Parent‘ bezeichnet den Index seiner übergeordneten Kontur. Wenn eine solche Kontur existiert nicht, wird dieses Feld als -1 angenommen. Wird die Aufgabe unserer Arbeit betrachtet, ist dieser Parameter nicht relevant, da nur eine der Konturen der Kreise benötigt wird. Dieser Parameter wird deswegen nicht weiter besprochen und berücksichtigt.

FindContours() hat insgesamt sechs Parameter. Einer davon ist das Eingabeparameter ‚image‘, das ein 8-Bit-Einkanalbild fordert. Pixeln ungleich Null in dem Eingabebild werden als Einsen behandelt. Nullpixeln bleiben Nullen, daher wird das Bild als binär behandelt. Die andere Parameter dieser Funktion sind ‚contours‘, ‚hierarchy‘, ‚mode‘, ‚method‘ und ‚offset‘, wobei ‚contours‘ und ‚hierarchy‘ Ausgaben sind. Der Parameter ‚mode‘ legt den Modus des Konturabrufs. In opencv 4.7.0 gibt es vier Kontur-Abruf-Modi: RETR\_EXTERNAL, RETR\_CCOMP, RETR\_LIST und RETR\_TREE.

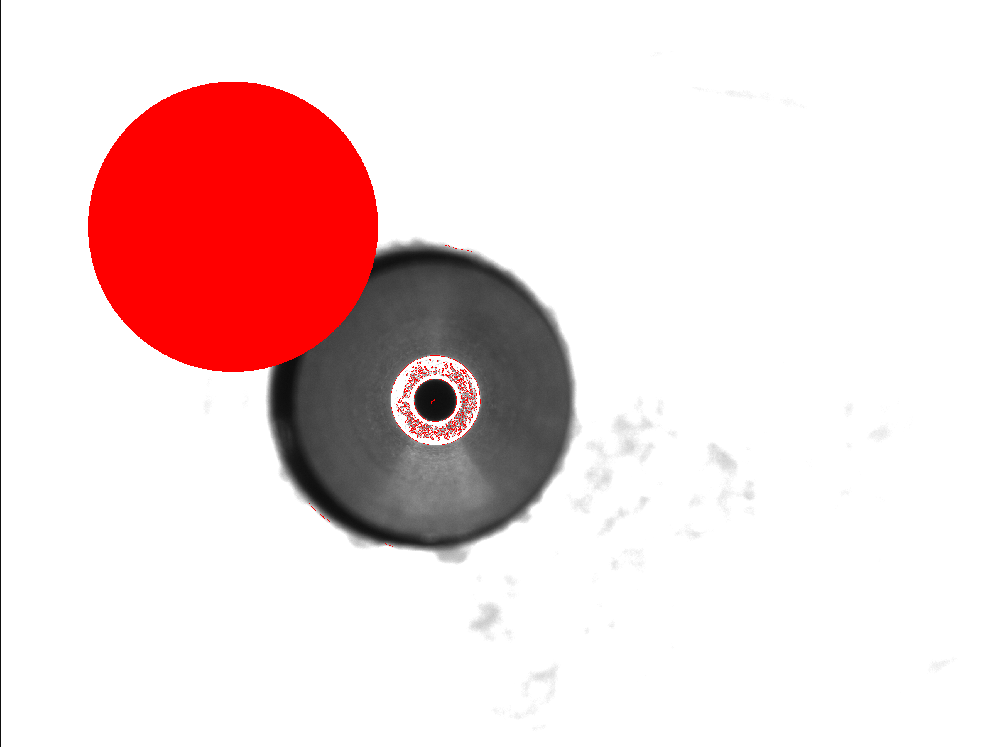
Der RETR\_EXTERNAL Modus gibt nur die extreme äußeren Konturen zurück und besitzt die schnelleste Geschwindugkeit. Im unseren Fall wird es nich in Betracht kommen. Der RETR\_CCOMP Modus ruft alle Konturen ab und ordnet sie in einer zweistufigen Hierarchie an. Die äußere Konturen des Objekts (d. h. seine Grenze) werden in Hierarchie 1 platziert. Und die Konturen der Löcher im Objekt, bzw. die innere Konturen (sofern vorhanden) werden in Hierarchie 2 platziert. Wenn sich darin ein Objekt befindet, wird dessen Kontur wieder in Hierarchie 1 platziert. Und wenn es darin ein Loch gibt, wird es wiede in Hierarchie 2 platziert und so weiter. Der RETR\_LIST Modus ruft alle Konturen ab, ohne hierarchische Beziehungen herzustellen， d. h. alle Konturen gehoeren zur gleichen Hierarchieebene. Hier ist also der 3. und 4. Term im Hierarchie-Array immer -1. Wie der RETR\_LIST Modus ruft der RETR\_TREE Modus alle Konturen ab. Außerdem erstellt es eine vollständige hierarchische Beziehungen und besitzt es deswegen die langsamste Geschwindigkeit. Wie vorher gesprochen, die Hierarchie der Konturen eines Bildes ist nicht relevant in unserer Aufgabe und deshalb passen alle letzten drei modi zu dieser Aufgabe. Für eine geringere Menge der Datenverarbeitung wird in dieser Aufgabe der Modus RETR\_LIST benutzt.

Der Parameter ‘method’legt die Methode zur Konturnäherung fest. Die vorhandenen Konturnäherungsmethoden sind: CHAIN\_APPROX\_NONE, CHAIN\_APPROX\_SIMPLE, CHAIN\_APPROX\_TC89\_L1, CHAIN\_APPROX\_TC89\_KCOS.

Die Methode CHAIN\_APPROX\_NONE speichert alle Konturpunkte da hier keine Konturnäherungsmethoden verwendet werden. Unten zeigt es die drei Versuche einschließlich den entsprechenden Daten mit diesem Parameter.

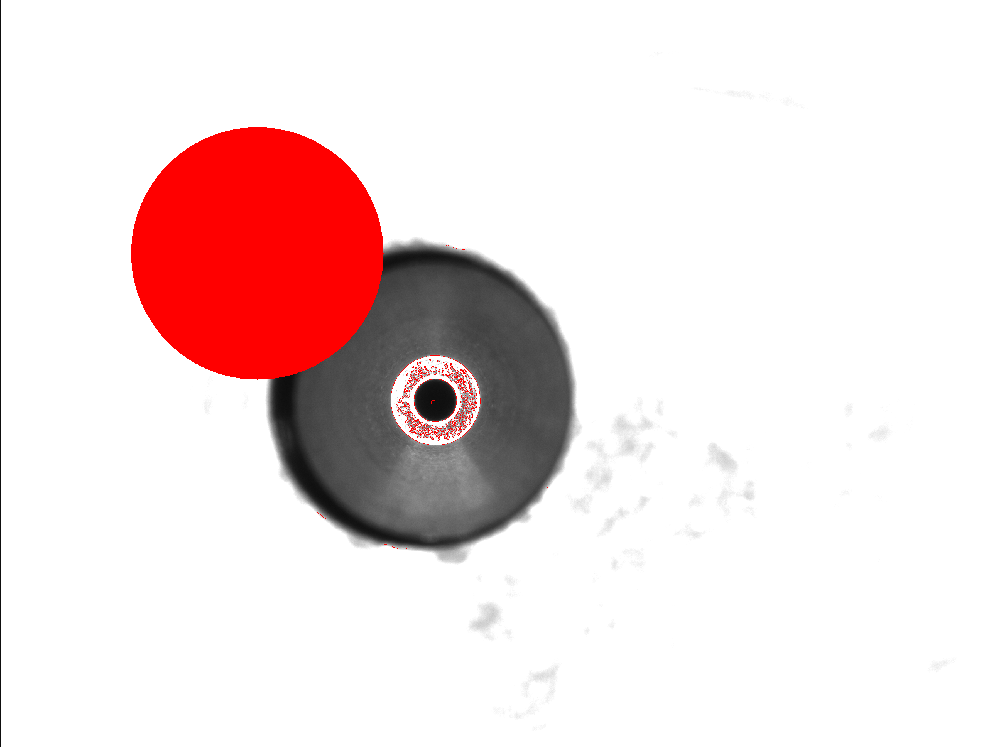
  
  
  
Vorhergesagter Mittelpunkt der roten Platte: (624.6164545602063,452.27858243524656)

Position des Vakuumgreifer relative zur roten Platte ist: (1101.634022135515 , -1292.6441743065202)  
erster mittelpunkt: (1125.3591918945312 , 1039.8441162109375)  
Die Abweichung ist: 1698.3893782331124 µm



Vorhergesagter Mittelpunkt der roten Platte: (616.3354675290558,461.19444695659007)

Position des Vakuumgreifer relative zur roten Platte ist: (1119.836080322796 , -1272.963476461127)  
Erster detektierte Mittelpunkt: (1125.3518676757812, 1039.814208984375)  
Abweichung: 1695.4258636686895 µm

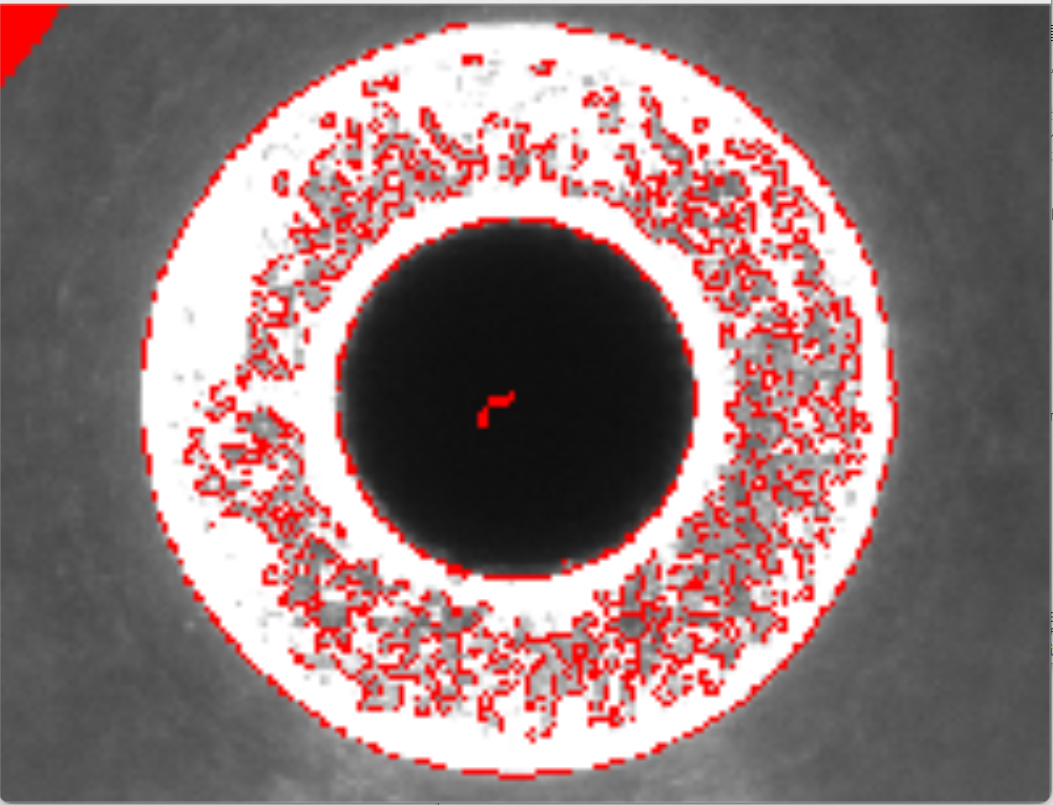


Vorhergesagter Mittelpunkt der roten Platte: (617.9553440320128,459.4285765248019)

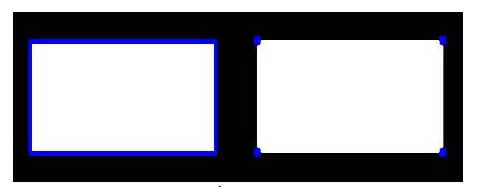
Position des Vakuumgreifer relative zur roten Platte ist: (1116.1449228170718 , -1276.8341580126234)  
Erster detektierte Mittelpunkt: (1125.2939453125 , 1039.8077392578125)  
Abweichung: 1695.9024016134101 µm

Durch die Beobachtung dieser drei Bilder und die entsprechen Daten ist es offensichtlich, dass es immer zu einem sclechten Ergebnis führt. Der vorhergesagter Mittelpunkt der roten Platte bzw. interpolierte Kreis liegt ganz weit zu der realen Position und die daraus resultierende kalibrierte Postion des Vakuumgreifers ist ebenso falsch.

Wird das Bild der ersten Bild vergrößert(siehe Abbildung 几), ist es klar zu sehen, dass die detektierte Konturpunkte des inneren Kreises nicht kontinuierlich ist, was zu einer möglichen falschen Interpolation des inneren Kreises führt. Dies dann führt zu einer falschen Interpolation des vorhergesagten Kreises.



Die Methode CHAIN\_APPROX\_SIMPLE entfernt alle überflüssigen Punkte und komprimiert die Kontur. Wie in der Abbildung 几 dargestellt wird die Kontur eines Rechtecks mit CHAIN\_APPROX\_NONE und CHAIN\_APPROX\_SIMPLE detektiert. Es ist offensichtlich zu sehen，dass nur die vier Endpunkte benötigt werden um ein Rechteck zu beschreiben. Dadurch wird der Speicherplatz erheblich gespart.

Detektierte Konturpunkte mit CHAIN\_APPROX\_NONE(links) und CHAIN\_APPROX\_SIMPLE(rechts)

Unten sind 。。。没图

草

Die Methode CHAIN\_APPROX\_TC89\_L1 und CHAIN\_APPROX\_TC89\_KCOS sind verwendet, wenn die Formen gekrümmt und keine einfachen Polygone sind, wobei Berechnungen mit höherer Präzision erfordern. In unserem Fall ist die zu detektierte Form eine einfache Ellipse, wobei die Methode CHAIN\_APPROX\_NONE, CHAIN\_APPROX\_SIMPLE schon reichen,deswegen werden diese zwei Methoden nicht weiter betrachtet.

Deswegen sind diese zwei Methoden in unserem Fall nicht geeignet.

https://cvexplained.wordpress.com/2020/06/03/finding-and-drawing-contours/

https://docs.opencv.org/4.7.0/d9/d8b/tutorial\_py\_contours\_hierarchy.html

https://docs.opencv.org/4.7.0/d3/dc0/group\_\_imgproc\_\_shape.html

none 和simple 对比

chain simple 举例子图用这个：

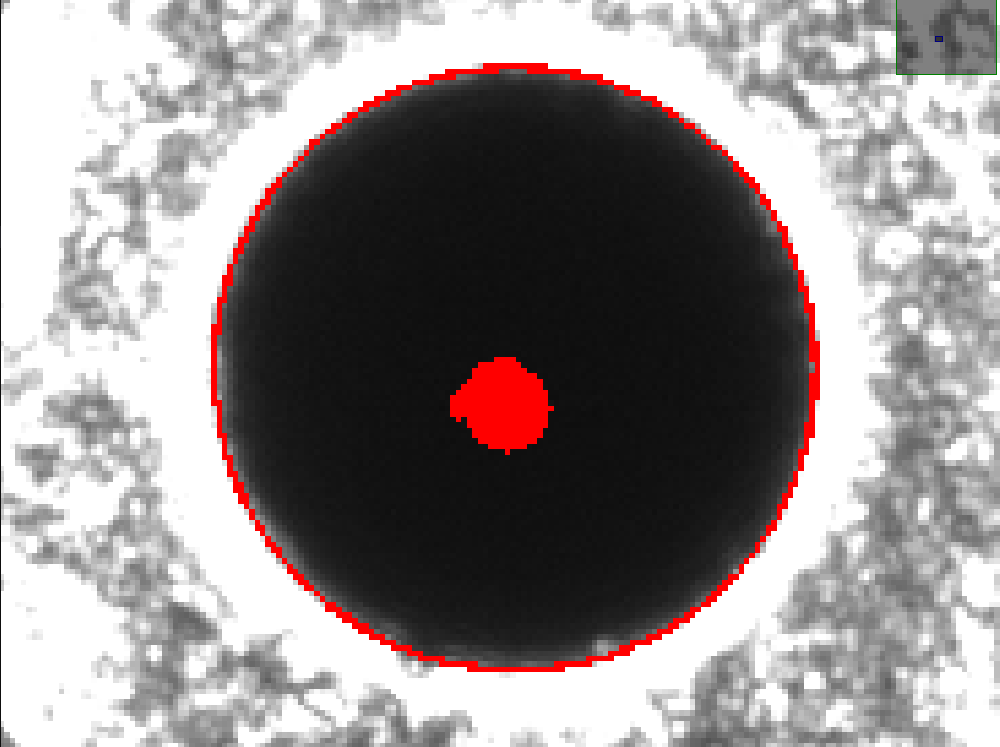
https://docs.opencv.org/4.7.0/d4/d73/tutorial\_py\_contours\_begin.html

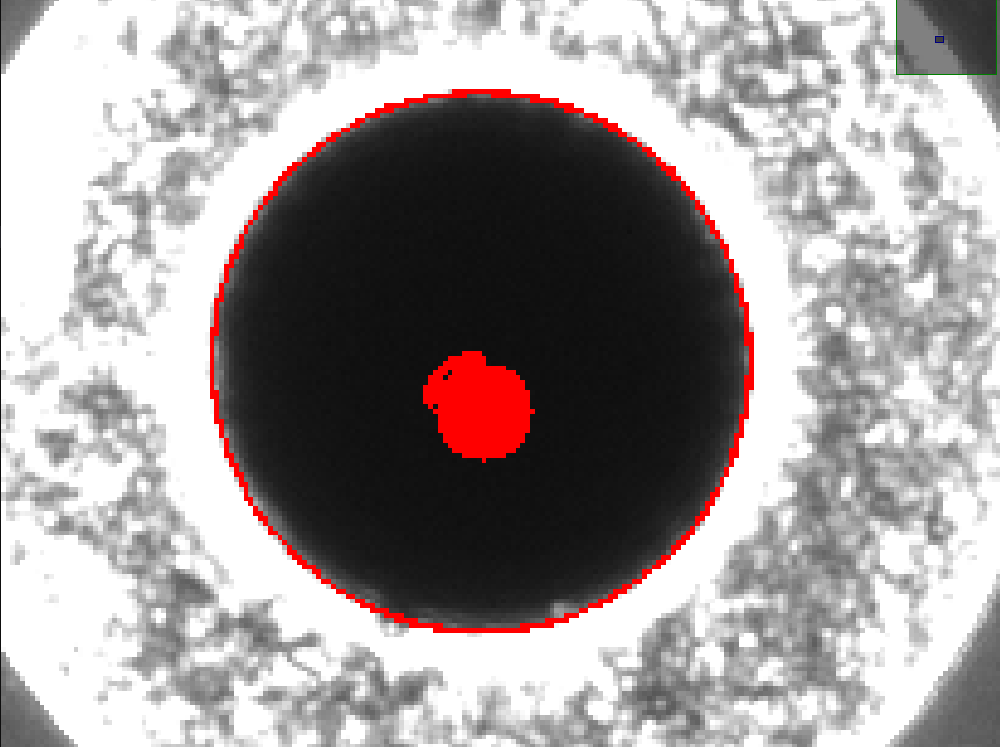
Der letzte Parameter von findContours() ist ‚Offset‘. Er ist ein optionaler Parameter, mitdem jeder Konturpunkt verschoben werden kann. Es ist verwendet um die Konturen, die aus dem Bild-ROI extrahiert werden, in dem Kontext des gesamten Bild zu analysieren. Dieser Parameter ist für diese Aufgabe nicht relevant, deswegen wird er in dieser Arbeit nicht weiter berücksichtigt.

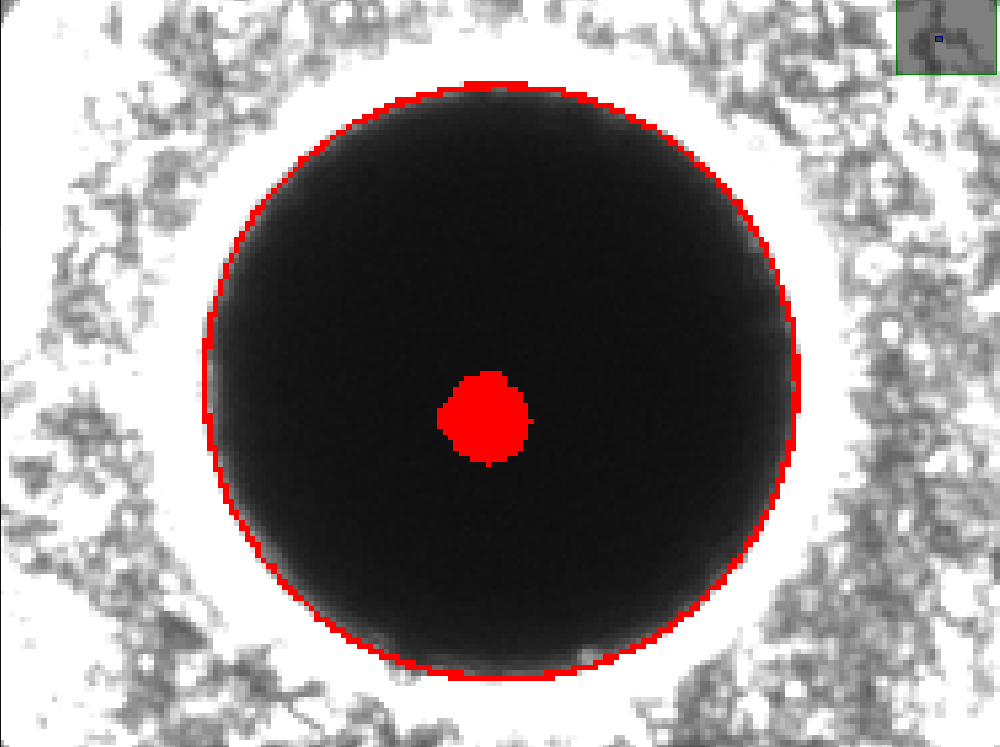
5. fitellips Direct

In OpenCV gibt es insgesamt drei Funktionen um eine Ellipse zu detektieren: fitEllipse(), fitEllipseAMS() und fitEllipseDirect(). Alle diese drei Methoden berechnen die Ellipse, die zu einer Reihe von 2D-Punkten passt. Der Unterschied liegt an die verwendeten Methoden. fitEllipse() rechnet die Algebraische Distanz, fitEllipseAMS() benutzt die Methode ‚Approximate Mean Square‘, und fitEllipseDirect() benutzt die Methode ‚Direct Least Square‘. Wegen der ganz kleinen Differenz der detektierten Ellipse zwischen diesen drei Funktionen, ist enie weitere Forschung zu diesen Methoden von weniger Bedeutung. Es wird deshalb nur die Auswirkungen dieser drei Methoden betrachtet und die Diskussion ihrer Algorithmen wird in dieser Arbeit verzichtet.

Unten sind







缺retval的数据，用数据说明三个方法之间的差距很小，用interpoliert 圆说明产生的效果。

ausserdem 还有几个小的函数 画圆。。。

先讲理论，再讲代码，讲代码的时候也讲参数。

讲上面的 ‚parameter‘

gauss里说，之前的sobel算子也是convolution 卷积块

也许写一下工作的流程： 用什么软件开什么灯之类的

Xacro: Zai zhe zhong sixiang de jichushang (basierend auf diese Gedanke,wird die Programmierung…)

Fehler:

1.

2.

3.

4. systematisch Fehler :其他部件的误差

我们还用了desired position，这样就每次一个方向转，也许可以改进成不用desired，直接来回并多次旋转看结果

本文用了fitellipse， 本应该进行圆补偿，但是由于检测到的椭圆的不确定性，所以没有进行。

展望一下的话最好是要进行的

1. Vgl. https://discourse.ros.org/t/real2sim-realistic-simulation-environments/19857 [↑](#footnote-ref-2)
2. [↑](#footnote-ref-3)