R2M2:
Roboter-Roboter-Mensch-Mimik

Abschlussdokumentation

Bachelor - Projekt Systementwicklung

Oktober 12, 2018

Inhaltsverzeichnis

1 Projektidee - der *Coztbote*

Cozmo ist ein programmierbarer Roboter der Firma Anki[1], der unter anderem über eine Kamera, künstliche Intelligenz und einen kleinen Hebearm zum Aufheben von mitgelieferten Würfeln verfügt.

Figure 1: Cozmo Roboter

Unsere ursprüngliche Projektidee war die Simulation einer Stadt, in der Postpakete von Cozmo - dem *Coztboten* - an die richtigen Personen ausgeliefert werden müssen. Die Strecken, auf denen sich Cozmo bewegen kann, werden durch schwarzes Klebeband auf möglichst hellem Untergrund abgeklebt, sodass sich der Roboter daran orientieren und der Strecke folgen kann. Dadurch ist es möglich, ein kleines Straßennetz mit verschiedenen Kreuzungsarten und Endpunkten aufzubauen. Am Ende einer Straße ist das Haus einer Person bzw. die Person selbst, die als potentieller Empfänger eines Pakets in Frage kommt. Der Coztbote soll dann prüfen, ob er hier richtig ist und das Paket entweder zustellen oder es bei den übrigen Endpunkten versuchen. Die zu liefernden Pakete holt sich der Coztbote in der Paketstation, die sich an einem besonderen Endpunkt des Straßennetzes befindet.

2 Projektziele

Cozmo soll ein Straßennetz erkennen und abfahren können. Er hält dabei die Spur und kann logisch auf Kreuzungen reagieren. Außerdem muss er in der Lage, sein Pakete (Würfel) aufzuheben und diese an die dafür vorgesehenen Endpunkte zu bringen. Cozmo muss das Paket dem richtigen Besitzer zuordnen. Sollte kein Besitzer gefunden werden, wird das Paket zurückgebracht.

2.1 Bonusziele

Für einen möglichen Zeitpuffer wurden außerdem Bonusziele erarbeitet. Bonusziele waren die Erkennung von Straßenschildern und die Interaktion mit einem zweiten Cozmo auf dem Straßennetz. Beispiele für diese Interaktionen sind z.B. eine Paketübergabe oder ein Ausweichmanöver.

3 Umsetzung

3.1 Architektur

Der Python-Code, der ausgeführt werden soll, läuft auf einem PC oder Laptop, das mit einem Mobilgerät verbunden ist. Auf diesem muss die Cozmo App installiert sein und im SDK Modus laufen, die ihrerseits mit dem Cozmo Roboter über dessen WLAN-Netz kommuniziert.

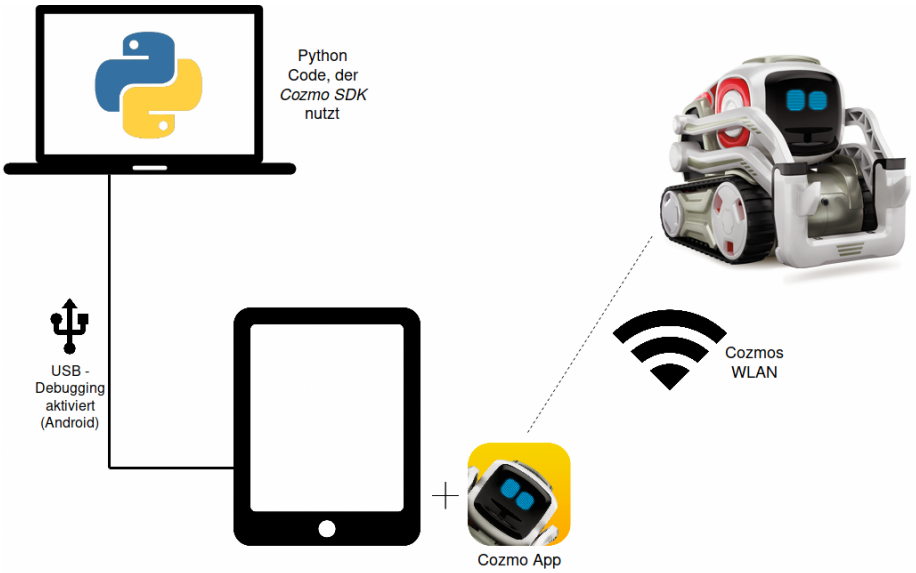

Figure 2: Veranschaulichung des Hardware-Aufbaus

3.2 Module

Es gibt verschiedene Teilfunktionen, die wir für Cozmo programmiert haben, die aber nicht alle gleichzeitig aktiv sein müssen oder sogar dürfen. Diese werden im Folgenden als **Module** bezeichnet (Beispielsweise bedarf es in der Paketstation keiner Erkennung von Kreuzungen, näheres hierzu im Abschnitt 3.3).

Jedes dieser Module basiert auf Bildverarbeitung und analysiert die aktuellen Frames, die der Roboter sieht. Bevor der aktuelle Frame an das aktive Modul bzw. die aktiven Module weitergeleitet wird, wird folgende Vorverarbeitung des Frames durchgeführt, um Störungen zu entfernen:
Die Cozmo Kamera nimmt ein Graubild auf, welches wir in ein Binärbild umwandeln. Mittels Erosion und Dilatation wird Rauschen aus dem Binärbild entfernt. Die Schwarz-Nachbarschaft mit dem größten Flächeninhalt wird dann auf ein neues weißes Bild extrahiert, sodass im Normalfall nur noch die Fahrbahn erhalten bleibt.

3.2.1 Spurhaltung

Damit Cozmo auf der Spur bleibt, wird immer berechnet, wie stark nach links oder rechts korrigiert werden muss. Dafür werden die unteren zwei Drittel des aktuellen Frames verwendet, da in diesem Bildausschnitt der relevante Teil der Fahrbahn sichtbar ist. Oberhalb ist die Fahrbahn mehr als 20cm von Cozmo entfernt und ist daher für die Berechnung der Korrektur irrelevant. Der übrige Teil des Bildes wird in drei Bereiche aufgeteilt.

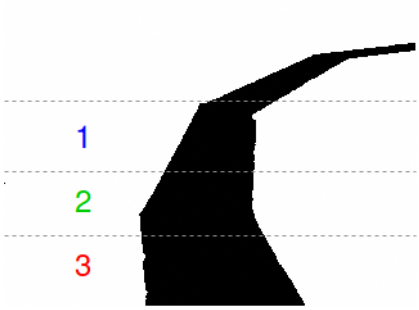

Figure 3: Veranschaulichung der Frameaufteilung

In jedem dieser drei Bereiche wird die geometrische Mitte seiner schwarzen Pixel berechnet. Dadurch erhalten wir drei Punkte die in ihrem Bildausschnitt auf dem ungefähren horizontalen Mittelpunkt der Fahrbahn liegen. Primär wird der oberste der drei Punkte zur Berechnung verwendet. Sollte dieser Punkt zum Beispiel in scharfen Kurven nicht erkannt werden, wird der mittlere Punkt verwendet. Ist dieser auch nicht sichtbar, wird nach dem untersten Punkt navigiert. Abhängig von der Position des Punktes findet eine Korrektur nach links oder rechts statt. Je weiter der Punkt von der Bildmitte entfernt ist, desto stärker die Korrektur. Kann keiner der drei Punkte erkannt werden, wird der letzte Korrekturwert beibehalten. Dies ermöglicht Cozmo die Fahrbahn beim Abkommen von der Fahrbahn diese wieder zu finden.

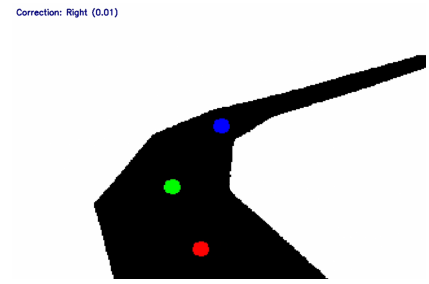

Figure 4: Berechnung der Navigationspunkte

3.2.2 Kreuzungserkennung

Zur Erkennung von Kreuzungen wird das Ausgangsbild zunächst auf einen kleineren Bereich zugeschnitten, um nur den Bereich, in dem sich das Kreuzungssegment befindet, zu verarbeiten. Der kleinere Bildbereich besteht aus weniger Pixeln und ist damit auch weniger rechenintensiv in der Bildverarbeitung. Jeder Frame der Kamera wird auf Kreuzungssegmente untersucht. Dies geschieht wie folgt:

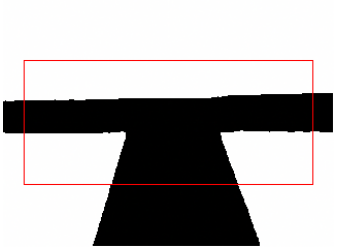

Figure 5: Bildausschnitt

1. Aus dem Bildausschnitt wird jede zehnte Pixelreihe in ein Array extrahiert
2. Jedes Array wird mittels Run-Length-Encoding komprimiert.
3. Zu kurze Musterwechsel (vermutlich Noise) werden entfernt.
4. Gleiche, aufeinanderfolgende Muster werden zusammengefasst.
5. Die am seltensten vorkommenden Mustergruppen werden entfernt, sodass nur drei Muster verbleiben.
6. Anhand dieser drei Muster wird das Kreuzungssegment identifiziert.
7. Wird die gleiche Kreuzungsart auf zwei konsekutiven Frames erkannt, wird das dazugehörige Verhalten aufgerufen.

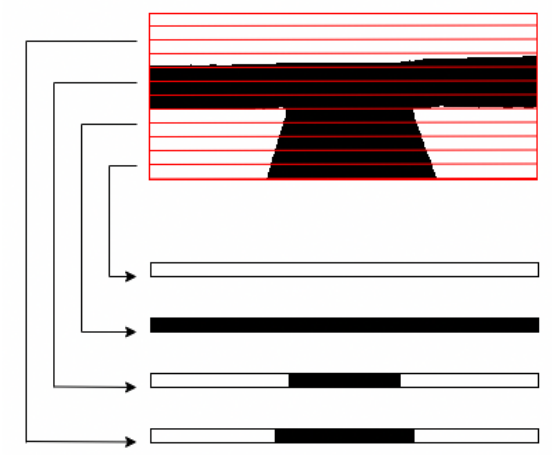

Figure 6: Pixelreihen die auf Muster untersucht werden, Anzahl dieser ist variabel

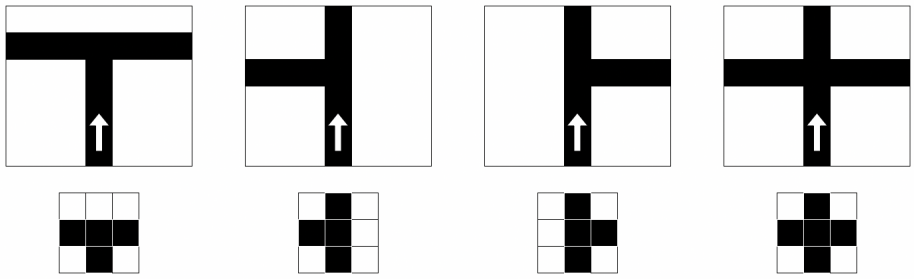

Figure 7: Kreuzungstypen und zugehörige Muster

Navigation
Damit Cozmo erfolgreich durch das Straßennetz navigieren kann, muss er vordefinierte Routen speichern und abfahren können. Dafür ist eine interne Liste aller möglichen Routen, bestehend aus einer Abfolge von den Operationen *links*, *rechts* und *geradeaus* angegeben, welche iterativ abgearbeitet werden.

3.2.3 Schilderkennung

Um weitere Funktionalitäten auszulösen, benötigten wir eine Form der Markierung. Im Fall des *Coztboten* haben wir uns für Straßenschilder entschieden, welche aus gleich großen Rechtecken bestehen. Diese Rechtecke werden symmetrisch links und rechts von der Fahrbahn angebracht. Je nach Anzahl der erkannten Rechtecke werden verschiedene Aktionen, wie zum Beispiel Anhalten oder Wenden, ausgeführt. Mit dem Ziel, diese Schilder erkennen zu können, haben wir uns für folgende Vorgehensweise entschieden.

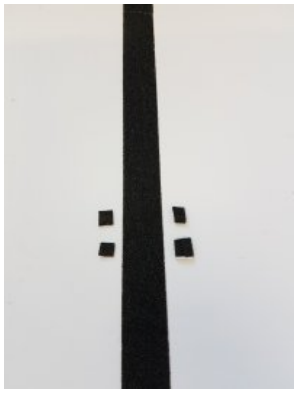

Figure 8: Beispiel für Schild neben der Fahrbahn

Jeder Kamera-Frame wird binarisiert und invertiert, um die Erkennung der Konturen zu ermöglichen. Störungen werden entfernt. Es sollen nur Schilder erkannt werden, die sich direkt vor Cozmo befinden. Dazu wird das Bild so zugeschnitten, dass nur die unteren zwei Drittel des Bildes betrachtet werden.

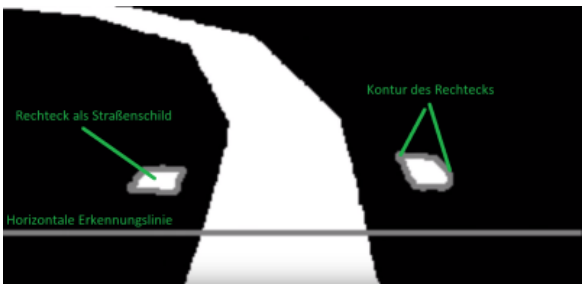

Figure 9: Analyse eines Frames

Des Weiteren wird eine horizontale Linie eingeführt, welche auf der Höhe von einem Sechstel des Bildes positioniert ist. Damit ein Schild erst erkannt wird, wenn es sich direkt vor Cozmo befindet, werden Rechtecke erst gewertet, wenn sich mindestens eins unterhalb dieser Horizontalen befindet. Mit einem Algorithmus der OpenCV2-Bibliothek[2] werden die Rechtecke als Konturen erkannt. Dabei werden nur Konturen zwischen einer gewissen Minimal- und Maximalgröße gewertet. Wird ein Schild erkannt, wird eine Abklingzeit eingeleitet, welche verhindert, dass Schilder in den nächsten Sekunden fälschlicherweise erneut oder falsch erkannt werden. Nach jedem Kamera-Frame wird geprüft, ob die Abklingzeit vorüber ist, und eventuell zurückgesetzt. Der Zustand der Schilderkennung wird dabei in einem Vorschaufenster angezeigt. Konturen sowie die Horizontale werden farblich hervorgehoben.

Insgesamt sechs erkannte Rechtecke stellen hierbei ein Wende-Schild dar. Um eine Paketstation zu markieren werden zwei, für einen Empfänger, vier Rechtecke benutzt.

3.2.4 Verhalten in der Paketstation

Das Verhalten in der Paketstation beinhaltet sowohl das Zurückkommen ohne Paket als auch mit Paket. In diesem Abschnitt werden diese Verhaltensweisen genauer beschrieben und die Unterschiede verdeutlicht.

Betreten ohne Paket
Sobald Cozmo das Paketstation-Straßenschild das erste Mal sieht und kein Paket bei sich trägt, wird das Programm zum Einsammeln eines Pakets gestartet. Dieses beginnt mit dem Suchen des Pakets. Sollte Cozmo das Paket nicht auf Anhieb finden, dreht er sich und sucht dort (im neuen Sichtbereich) weiter. Wenn er ein Paket findet, fährt er auf dieses zu und passt seinen Weg dahingehend an, dass er es aufheben kann. Das Aufheben des Pakets funktioniert nicht immer (falschen Anpassungen geschuldet), daher muss Cozmo nach einem gescheiterten Versuch eine kleine Strecke rückwärtsfahren und es erneut versuchen, bis es funktioniert.
Wenn Cozmo das Paket erfolgreich aufgehoben hat, weiß er, wer dessen Empfänger ist und begibt sich auf den Weg zur Auslieferung. Er folgt nun dem nächsten erkannten Streckenabschnitt. Sollte Cozmo nicht Richtung Ausgang fahren, befindet sich am Ende der Paketstation ein Wendeschild, welches dafür sorgt,

dass Cozmo um 180 Grad dreht und Richtung Ausgang fährt.

Betreten mit Paket
Sollte Cozmo die Paketstation mit einem Paket betreten, gibt es zwei Fälle zu betrachten. Entweder war Cozmo bei jedem Haus und kein Besitzer passte oder er hat sich verfahren. Cozmo ignoriert das Paketstation-Straßenschild bis er entweder bei allen Endpunkten war oder das Paket ausgeliefert hat.
War er bei allen Endpunkten des Straßennetzes, hat den richtigen Empfänger aber nicht gefunden, so stellt er das Paket beim Betreten der Paketstation ab und nimmt ein neues auf. Die hier verwendete Routine ist bis auf das vorherige Abstellen des alten Pakets die gleiche wie die ohne Paket.

3.3 Zusammenspiel der verschiedenen Module

Die einzelnen Funktionalitäten wurden meist isoliert als atomares Modul entwickelt. Schritt für Schritt wurden dann die Module integriert. Um das Zusammenspiel der zunächst eigenständigen Funktionalitäten zu gewährleisten, mussten Anpassungen am Kerngerüst des Programms gemacht werden. Durch die Einführung von Controller-Klassen ließen sich Zustände, ähnlich denen eines Zustandsautomats, dynamisch speichern. Folglich konnten einzelne Module Prüfungen auf diese Zustände durchführen, um zu entscheiden, welche Module aktiv sein müssen und welche nicht. Im Folgenden wird der Ablauf der Zustände beschrieben.

Zum Start des Programms wird Cozmo vor einer Paketstation platziert. Das Programm befindet sich anfangs im Zustand der Spurhaltung und Schilderkennung. Er fährt in die Paketstation hinein, wobei ein Paketstation-Straßenschild erkannt wird. Er wechselt durch Erkennen des Schildes in den Paket-Aufnehmen-Zustand. Hierfür werden die vorher benutzten Funktionalitäten wie das Straßen- und Schilderkennen ausgeschaltet. Durch Funktionalitäten des Cozmo-SDK[3] hebt Cozmo nun das Paket auf. Nach dem Aufheben wird die Straßen- und Schilderkennung wieder eingeschaltet. Er verlässt die Paketstation und befindet sich durch das erneute Erkennen des Paketstation-Schilds im Zustand der Zustellung. Die Route zum ersten potentiellen Empfänger wird abgefahren. Erreicht er den Empfänger, wird ein Empfänger-Straßenschild eingelesen. Straßen- und Spurerkennung werden ausgeschaltet und er ist jetzt im Zustand der Gesichtserkennung. Je nachdem ob der richtige Empfänger erkannt wurde, wird die Route zurück zur Paketstation oder zum nächsten Empfänger gesetzt. Er nimmt die Straßen- und Schilderkennung wieder auf. Dieses Verhalten wiederholt sich, bis das Programm manuell angehalten wird.

3.4 Programmierrichtlinien

Um eine gute Wartbarkeit des Projekts zu fördern, wurde während der Entwicklungsphase eine objektorientierte Struktur verfolgt sowie die offiziellen PEP8 Programmierrichtlinien eingehalten.

3.5 Verwendete Sprachen und Frameworks

Als verwendete Sprache kam Python zum Einsatz, da die Firma Anki[1] die *Cozmo SDK*[3] nur für Python bereitstellt. Zudem wurde zur Bildverarbeitung die OpenCV Bibliothek[2] verwendet.

4 Endergebnis

Cozmo beginnt mit der Suche und dem Aufheben eines Pakets in der Paketstation. Anschließend verlässt er die Paketstation und navigiert eigenständig über das von uns erstellte Straßennetz. Dabei beachtet und interpretiert er Markierungen neben der Fahrbahn um das Straßennetz nach dem richtigen Empfänger abzusuchen. Wenn kein passender Empfänger gefunden wurde, bringt er das Paket zurück. Sollte der Empfänger angetroffen werden, wird das Paket zugestellt. In beiden Fällen wird ein neues Paket aus der Paketstation geholt.

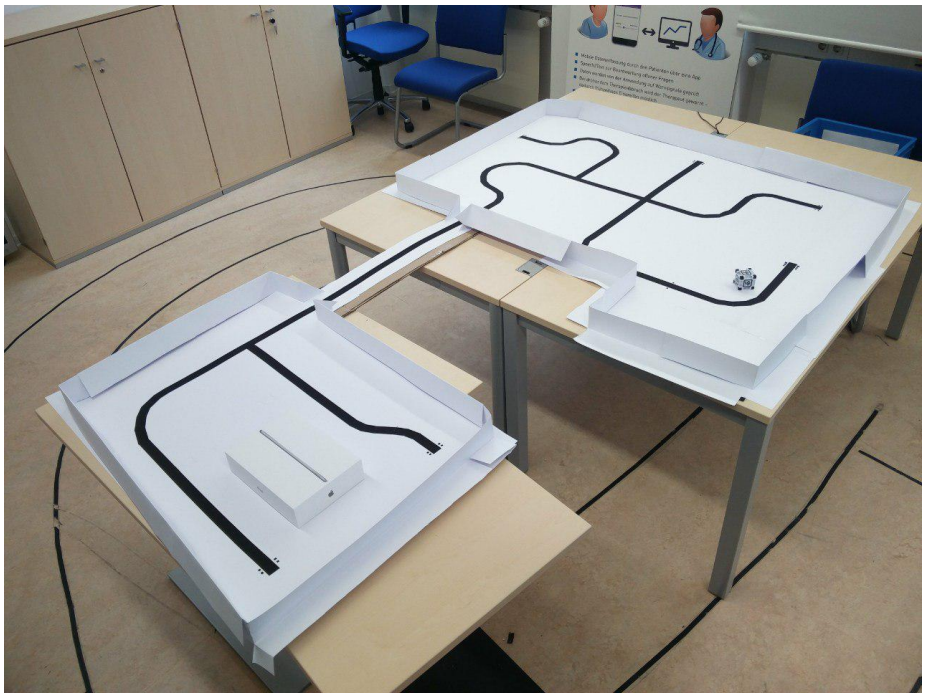

Figure 10: Fertiges Straßennetz mit Brücke

5 Installation

5.1 Benötigte Hardware

- PC oder Laptop
- Cozmo Roboter
- WLAN-fähiges Mobilgerät (z.B. Smartphone/Tablet), für das die Cozmo App verfügbar ist
    - USB-Verbindungskabel

5.2 Benötigte Materialien

- Straßennetz bzw. Strecke, auf dem bzw. der Cozmo sich bewegen kann

5.3 Benötigte Software

- Auf dem PC/Laptop
    - Python 3 [4]
    - pipenv [5] (alternativ können die Abhängigkeiten auch manuell über *pip* installiert werden)
    - Python Code aus unserem Repository [6]
- Auf dem Mobilgerät
    - Cozmo App

5.4 Schritte

1. Unseren Python Code aus dem *gruppe4-cozmo* Repository[6] herunterladen
2. In das Verzeichnis mit der *Pipfile* navigieren
3. Abhängigkeiten durch Ausführen des Befehls *pipenv install* installieren (dies setzt eine virtuelle Umgebung nur für dieses Projekt auf und installiert darin alle benötigten Pakete)
4. Mobilgerät mit dem WLAN-Netz von Cozmo verbinden (Cozmo zeigt WLAN-Passwort beim Starten auf seinem Display)
5. Am Mobilgerät USB-Debugging aktivieren
6. Mobilgerät mit PC verbinden
7. Cozmo App starten und den SDK Modus aktivieren
8. Cozmo auf der Strecke platzieren
9. *core.py* ausführen

6 Retrospektive

6.1 Ausblick

Mit Fertigstellung des Projekts kamen viele Erweiterungsmöglichkeiten auf. Darunter fällt auch die schon in den Bonuszielen bereits angesprochene Interaktion mit einem zweiten Cozmo auf. Hierbei stellt sich die Frage, was geschieht, wenn zwei Cozmos sich auf der Straße begegnen. Die Cozmos könnten sich gegenseitig erkennen und daraufhin reagieren. Mögliche Szenarien wären dann, dass die Cozmos sich gegenseitig ausweichen oder das Paket übergeben. Auch könnten sie sich an festgelegte Verkehrsregeln halten. Ein Beispiel für eine solche Regel wäre, dass ein Cozmo mit einem Paket immer Vorfahrt hat. Die simulierte Stadt könnte auch mit zusätzlichen Bezirken erweitert werden, die über eine Übergabestelle miteinander verbunden sind. Zwei Cozmos würden dabei jeweils in ihrem Bezirk arbeiten und Pakete, die nicht in ihrem Zustellungsgebiet liegen, an die Übergabestelle bringen, wo der andere Cozmo sie abholen könnte. Zudem stellt sich die Frage, wie Cozmo auf Hindernisse reagieren könnte, da auch in der realen Welt im Straßennetz immer wieder unvorhergesehene Blockaden entstehen. Cozmo müsste somit erkennen wenn die Straße blockiert ist und logisch nach vorliegendem Problem handeln. Auch bei den Paketen und Häusern kommen interessante Erweiterungen in den Sinn. Statt der Würfel und Personen könnten auch eigene Pakete und Häuser gebaut werden. Die Verknüpfung könnte beispielsweise über unär-kodierte Zahlen oder mithilfe einer Farberkennung realisiert werden. Eine Priorisierung der Pakete wäre auch denkbar. Derzeit können nur bereits eingespeicherte Personen ein Paket erhalten. Erweiterbar wäre dies, indem zuvor eine schnelle Zuordnung von Paketen und Personen gemacht wird. Bilder der Person könnten auch direkt auf den Würfel geklebt werden, um so eine Zuordnung zu erstellen. Cozmo würde dann in der Lage sein, völlig Fremden das richtige Paket zu liefern.

6.2 Aufgetretene Probleme

6.2.1 Verzögerungen beim Senden von Befehlen

Anfangs entstand eine größere Verzögerung zwischen der Ausführung des Programmcodes und der zugehörigen Umsetzung durch Cozmo. Begründet war dies zum einen im anfangs ineffizienten Programmcode, zum anderen darin, dass Cozmo bei niedrigem Akkustand eine geringere Bearbeitungsgeschwindigkeit hat. Beheben ließ sich dies durch eine effizientere Gestaltung des Programmcodes sowie durch Sicherstellen, dass der Akku des Cozmo nicht unter einen kritischen Wert fällt. Hierzu wurde im Vorschaufenster auf dem PC eine Akkustandsanzeige platziert, sodass der Stand ständig beobachtet werden kann.

6.2.2 Erkennung von Gabelungen

Ursprünglich waren neben Kreuzungen auch Gabelungen im Straßennetz geplant. Hier stießen wir allerdings auf das Problem, dass die Muster wie eine

bestimmte Gabelung erkannt wird unterschiedlich aussehen, je nachdem von welcher Seite Cozmo die Gabelung anfährt (spitze Winkel, stumpfe Winkel). Man hätte den Winkel, in dem Cozmo bei einer Gabelung abbiegt, jedes Mal anhand der Frames berechnen müssen, wohingegen dieser Aufwand bei Kreuzungen im 90 Grad Winkel entfällt. Aus diesem Grund entschieden wir uns für die Kreuzungen, wobei es dann auch blieb.

6.2.3 Abklingzeiten

Da viele Informationen schnell hintereinander ausgewertet wurden, kam es oft vor, dass Befehle fälschlicherweise doppelt ausgeführt wurden. Ein Beispiel hierfür wären die Straßenschilder. Während der Abarbeitung des Befehls welcher durch Erkennen des Straßenschildes ausgelöst wird, wird dasselbe Straßenschild erneut erkannt und dadurch der erwartete Befehl häufiger als gewünscht Ausgeführt. Um dieses Problem zu lösen, haben wir interne Abklingzeiten eingebaut, die dafür sorgen, dass nach Abhandeln eines Events die Abklingzeit ausgelöst wird, welche verhindert dass für eine gewisse Zeit neue Trigger erkannt werden.

6.2.4 Unzuverlässige Abarbeitung der Befehlskette

Ein weiteres Problem war die unzuverlässige Abarbeitung von Befehlen wie zum Beispiel das Aufheben von Blöcken, was nicht immer gelingen wollte. Das lag zum einen daran, dass er nicht im richtigen Winkel an den Block heran fuhr und somit seine Haken nicht in die Halterungen des Blocks glitten, zum anderen wurde nach einer gewissen Anzahl an Versuchen aufgehört, was zum Nicht-Aufheben des Würfels führte. Lösen konnten wir das Problem, indem wir nach einer vorgegebenen Anzahl an Versuchen Cozmo 10cm zurück fahren ließen und er es in einem neuen Winkel noch einmal versucht. Auf dieses Weise war er immer in der Lage, den Würfel aufzuheben.

6.2.5 Cozmo SDK Verständnis

Zu Beginn des Projekts gab es Verständnisprobleme mit der API des Cozmo. Die Dokumentation war unseres Erachtens nicht immer schlüssig und warf häufig Fragen auf. Durch Probieren und Arbeiten mit der API begannen wir, ein tieferes Verständnis der Arbeitsweise zu entwickeln und wurden zunehmend sicherer.

6.2.6 Fälschliche Kreuzungserkennung bei Kurven

Figure 11: Das Muster WSW-SW-WSW wird bei der Kreuzungserkennung einer nach links gedrehten T-Kreuzung zugeordnet.

Bei scharfen S-Kurven trat das Problem auf, dass eine Kreuzung erkannt wurde. Dies passierte, da die Fahrbahn an diesen Stellen teilweise außerhalb des aufgenommenen Bildbereiches lag und so das gleiche schwarz-weiß-Muster entstand, wie es bei einer Kreuzung der Fall ist. Um dem vorzubeugen, wird, wenn eine Kreuzung erkannt wird, überprüft, ob der oberste Navigationspunkt vorhanden ist und sich Recht mittig im Bild befindet. Ist der Navigationspunkt zu weit links oder rechts oder gar nicht vorhanden, wird die erkannte Kreuzung für ungültig erklärt und ignoriert.

6.2.7 Gesichtserkennung

Die Funktionalität der Gesichtserkennung aus der API des Cozmo funktionierte anfangs noch sehr dürftig. Die Ursache war, dass die Gesichtserkennung stark von der Belichtung des vorher eingespeicherten Gesichts lag. Wurde ein Gesicht mit Gegenlicht eingespeichert, war es später schwer ohne Gegenlicht erkannt zu werden. Beheben ließ sich dies, indem man für gleiche Lichtverhältnisse bei der Einspeicherung und beim Betrieb sorgt.

6.2.8 Zeichenerkennung

Bei der Straßenschilderkennung wurden häufig Rechtecke erkannt, obwohl kein Straßenschild zu sehen war. Die Ursache lag in der Erkennung von Rauschpixeln wie z.B. weiter entfernte Straßenabschnitte oder ungewünschte Störungen wie Schattenwurf entstanden. Zwei Mechanismen sorgten für Abhilfe. Da Straßenschilder stets symmetrisch aufgebaut sind, konnten Messungen mit einer ungeraden Anzahl von Rechtecken verworfen werden. Des Weiteren wurden eine Mindest- und eine Maximalgröße festgelegt. Diese Schwellwerte verhinderten, dass sowohl kleine Rauschpixel, als auch die Straße als Schild erkannt wurden.

7 Referenzen

[1] *Anki*. URL: https://www.anki.com/de-de (visited on 12/10/2018).

[2] *opencv/opencv*. URL: https://github.com/opencv/opencv (visited on 12/10/2018).

[3] *Cozmo SDK*. URL: https://www.anki.com/en-us/cozmo/SDK (visited on 12/10/2018).

[4] *Welcome to Python.org*. URL: https://www.python.org/.

[5] *Pipenv: Python Dev Workflow for Humans*. URL: https://pipenv.readthedocs.io/en/latest/ (visited on 12/10/2018).

[6] *GitLab Repository*. URL: https://gitlab.fbi.h-da.de/est/lab/pse-est/bpse_ws1819/gruppe4-cozmo (visited on 12/10/2018).