Benjamin Pötz BSc

Evaluierung und Funktionales Testen eines Fahrer-Assistenzsystems für den Einsatz in der Landwirtschaft

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Elektrotechnik-Wirtschaft

eingereicht an der

Technischen Universität Graz

Betreuer

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Axel Pinz

Institut für Elektrische Messtechnik und Sensorik

Graz, August 2020

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

KURZFASSUNG

Die Masterarbeit hat zum Ziel, einen „Software in the Loop“ Prüfstand zu entwickeln, mit dessen Hilfe ein bestehender bildgestützter Algorithmus zur Pflanzenreihen-Erkennung getestet und evaluiert werden kann. Um die Frage zu beantworten, ob man mithilfe von synthetischen Bildern Algorithmen evaluieren kann, wird eine virtuelle Umgebung in der „Unreal Engine“ erstellt. Erfasst werden diese synthetischen Bilder von einer virtuellen Kamera, welche ebenso im Zuge dieser Arbeit vorgestellt wird. Da im vorliegenden Algorithmus sowohl die „Lineare Regression“ als auch die „Stripe Analysis“ für die Approximation der Pflanzenreihen zur Verfügung stehen, wird evaluiert, welches der beiden Verfahren besser für die Detektion von Pflanzenreihen geeignet ist. Anschließend werden umfangreiche Simulationen durchgeführt, die Kameraeffekte, Farbabweichungen, geometrische Einflüsse sowie Unkraut und Wettereinflüsse untersuchen. Mithilfe von 17 Messreihen kann gezeigt werden, dass der Algorithmus in den meisten Simulationen, je nach Pflanzenabstand, mittlere Abweichungen zur Ideallinie zwischen 5 mm und 10 mm erreicht. Durch den Vergleich mit realen Bildern lässt sich schlussfolgern, dass diese Ergebnisse auch in der Realität haltbar sind und sich synthetische Bilder daher hervorragend für die Entwicklung und Evaluierung von bildgestützten Algorithmen eignen.

ABSTRACT

The aim of the master thesis is to develop a “Software in the Loop” test bench to test and evaluate an existing image-based algorithm for plant row recognition. To answer the question of whether algorithms can be evaluated using synthetic images, a virtual environment is created in the “Unreal Engine”. These synthetic images are captured by a virtual camera, which is also presented in the course of this work. Since both “linear regression” and “stripe analysis” are available for the approximation of the plant rows in the present algorithm, the two methods are evaluated to determine which is better suited for the detection of plant rows. Then, extensive simulations are carried out, which examine camera effects, color deviations, geometric influences, as well as weeds and weather influences. With the help of 17 series of measurements, it can be shown that, in most simulations, depending on the distance between plants, the algorithm achieves mean deviations from the ideal line between 5 mm and 10 mm. By comparing them with real images, it can be concluded that these results are also valid for real scenes. Synthetic images are, therefore, ideal for the development and evaluation of image-based algorithms.

ABKÜRZUNGSVERZEICHNIS

ABS … Absolutbetrag

ADAS … Advanced Driver Assistance Systems

API … application programming interface

BSON … Binary JavaScript Object Notation

CCD … charge-coupled device

CMOS … complementary metal-oxide-semiconductor

CTE … Cross Track Error

DGPS … differential global positioning system

ECU … electronic control unit

ExG … Excess green index

ExGR … ExG minus ExR vegetation index

ExR … Excess Red index

FAS … Fahrerassistenzsystem

FOV … Field of View

GPS … global positioning system

HiL … Hardware in the Loop

JSON … JavaScript Object Notation

LR … Lineare Regression

MiL … Model in the Loop

ROI … Region of Interest

ROS … Robot Operation System

RTK DGPS … Real Time Kinematic differential global positioning system

SA … Stripe Analysis

SIL … Software in the Loop

UE … Unreal Engine

VR … virtuelle Umgebung

HSV … hue saturation value

INHALTSVERZEICHNIS

KURZFASSUNG 1

ABSTRACT 1

ABKÜRZUNGSVERZEICHNIS 2

INHALTSVERZEICHNIS 3

1 EINLEITUNG 7

2 VERWANDTE ARBEITEN 10

3 ALTERNATIVE LÖSUNGSWEGE 11

4 DER PRÜFSTAND 14

4.1 FUNKTIONSPRINZIP 14
4.2 DER ALGORITHMUS 15
4.3 „ROBOT OPERATING SYSTEM“ 17
4.3.1 Aufbau 17
4.4 ERZEUGTE DATEN 18
4.5 APPROXIMATIONSVERFAHREN 20
4.5.1 „Lineare Regression“ 20
4.5.1.1 Lineare Regressionsmodell 21
4.5.1.2 Notation 21
4.5.1.3 Methode der kleinsten Fehlerquadrate 21
4.5.2 „Stripe Analysis“ 23
4.6 BEURTEILUNG DER SIMULATIONSERGEBNISSE 24
4.7 PRÄZISION 26

5 DIE VIRTUELLE UMGEBUNG 29

5.1 ALLGEMEIN 29
5.2 ANFORDERUNGEN 29
5.3 ERSTELLEN DER VIRTUELLEN UMGEBUNG 29
5.3.1 „Blueprints“ 29
5.3.1.1 Automatisiertes Platzieren von Objekten 30
5.3.1.2 Steuerung der Kamera 31
5.3.2 Plugins 31
5.3.2.1 „Sun Position Calculator“ 31
5.3.2.2 „ROS Integration“ 32
5.3.2.3 „URoboVision“ 32

5.3.3 Visual Effects 32
5.3.3.1 „Sky Light“ 32
5.3.3.2 „Directional Lights“ 33
5.3.3.3 „Sky Atmosphere“ 33
5.3.3.4 „Exponential Height Fog“ 33
5.3.4 Einschränkungen 33
5.3.4.1 Die Pflanzen 33
5.3.4.2 Der Boden 34
5.3.4.3 Fazit 34

6 DAS KAMERAMODELL 35

6.1 KAMERAMODELL DER „UNREAL ENGINE“ 35
6.2 KAMERAMODELL ROS 37
6.3 KAMERAEFFEKTE 38
6.3.1 Bewegungsunschärfe 38
6.3.2 Rauschen 39
6.4 REALE KAMERA INTEL D435 39

7 SIMULATIONEN 41

7.1 „LINEARE REGRESSION“ VS. „STRIPE ANALYSIS“ 42
7.1.1 Messreihe ohne Pflanzenoffset 42
7.1.1.1 Beschreibung 42
7.1.1.2 Ergebnisse 42
7.1.2 Messreihe mit Pflanzenoffset 44
7.1.2.1 Beschreibung 44
7.1.2.2 Ergebnisse 44
7.1.3 Messreihe mit Aussetzern in der Pflanzenreihe 45
7.1.3.1 Beschreibung 45
7.1.3.2 Ergebnisse 45
7.1.4 Messreihe mit unterschiedlichen Pflanzen 48
7.1.4.1 Beschreibung 48
7.1.4.2 Ergebnisse 48
7.1.5 Messreihe mit Unkraut 49
7.1.5.1 Beschreibung 49
7.1.5.2 Ergebnisse 49
7.1.6 Analyse der Approximationsverfahren 50
7.2 KAMERAEFFEKTE 54
7.2.1 Messreihe Bewegungsunschärfe 54
7.2.1.1 Beschreibung 54
7.2.1.2 Ergebnisse 55
7.2.2 Messreihe Bildrauschen 56

7.2.2.1 Beschreibung 56
7.2.2.2 Ergebnisse 56
7.2.3 Messreihe Linsenverschmutzung 57
7.2.3.1 Beschreibung 57
7.2.3.2 Ergebnisse 58
7.2.4 Analyse der Kameraeffekte 59
7.3 ROBUSTHEIT GEGENÜBER FARBABWEICHUNGEN 61
7.3.1 Messreihen Variation der Farbtemperatur 61
7.3.1.1 Beschreibung 61
7.3.1.2 Ergebnisse 62
7.3.2 Messreihe Farbsättigung der Pflanzen 64
7.3.2.1 Beschreibung 64
7.3.2.2 Ergebnisse 65
7.3.3 Messreihe zusätzliche Grünflächen 66
7.3.3.1 Beschreibung 66
7.3.3.2 Ergebnisse 67
7.3.4 Messreihen Sonnenstände 68
7.3.4.1 Beschreibung 68
7.3.4.2 Ergebnisse 69
7.3.5 Messreihe Penumbra 72
7.3.5.1 Beschreibung 72
7.3.5.2 Ergebnisse 73
7.3.6 Analyse der Farbabweichungen 74
7.4 ROBUSTHEIT GEGENÜBER GEOMETRISCHEN EINFLÜSSEN 77
7.4.1 Messreihe mit Kameraerschütterungen 77
7.4.1.1 Beschreibung 77
7.4.1.2 Ergebnisse 77
7.4.2 Messreihen Skalierung der Pflanzen 78
7.4.2.1 Beschreibung 78
7.4.2.2 Ergebnisse 79
7.4.3 Messreihen mit Rotationen der Kamera 81
7.4.3.1 Beschreibung 81
7.4.3.2 Ergebnisse 81
7.4.4 Messreihe Kurvenfahrt 85
7.4.4.1 Beschreibung 85
7.4.4.2 Ergebnisse 85
7.4.5 Analyse der geometrischen Einflüsse 86
7.5 ROBUSTHEIT GEGENÜBER UNKRAUT 88
7.5.1 Messreihe ohne „Blob reduction“ 88
7.5.1.1 Beschreibung 88

7.5.1.2 Ergebnisse 89
7.5.2 Messreihe mit „Blob reduction“ 90
7.5.2.1 Beschreibung 90
7.5.2.2 Ergebnisse 90
7.5.3 Messreihen mit 15 cm Pflanzenabstand 91
7.5.3.1 Beschreibung 91
7.5.3.2 Ergebnisse 91
7.5.4 Analyse Unkraut 93
7.6 ROBUSTHEIT GEGENÜBER WETTEREINFLÜSSEN 94
7.6.1 Messreihe Nebel 94
7.6.1.1 Beschreibung 94
7.6.1.2 Ergebnisse 95
7.6.2 Messreihe Wasserstände 96
7.6.2.1 Beschreibung 96
7.6.2.2 Ergebnisse 97
7.6.3 Analyse der Wettereinflüsse 98
8 ANALYSE DER SIMULATIONS - ERGEBNISSE 99
9 VERGLEICH MIT REALEN MESSUNGEN 101
9.1 REALE BILDER 101
9.2 AUSWERTUNG DER BILDER 103
9.3 SYNTHETISCHE BILDER 105
9.4 ERKENNTNISSE 106
10 ZUSAMMENFASSUNG UND AUSBLICK 107
11 ABBILDUNGSVERZEICHNIS 108
12 TABELLENVERZEICHNIS 111
13 LITERATURVERZEICHNIS 111

1 EINLEITUNG

Traktoren sind die Arbeitstiere der modernen Landwirtschaft. Durch Automatisierung dieser Maschinen ist es möglich, die Produktivität und die Sicherheit zu erhöhen und gleichzeitig in vielen Bereichen der Landwirtschaft die Kosten zu senken [1]. Insbesondere die mechanische Unkrautbekämpfung war bisher mit einem hohen Arbeitsaufwand verbunden. Bislang steuerte der Fahrer selbst die Maschine zur Unkrautbekämpfung, die seitlich von den Erntereihen geführt werden muss. Diese Aufgabe erfordert viel Konzentration und ist über einen längeren Zeitraum hindurch kaum aufrechtzuerhalten, da der Fahrer sowohl den Traktor als auch das sogenannte Implement steuern muss. Um diesen Aufwand zu reduzieren, wurden automatisierte Hackmaschinen entwickelt, welche in unmittelbarer Nähe der Kulturpflanzen arbeiten, ohne sie zu beschädigen. D.C. Slaughter et al. [2] beschreiben vier Kerntechnologien, die für die Entwicklung eines Robotersystems zur Unkrautbekämpfung nötig sind.

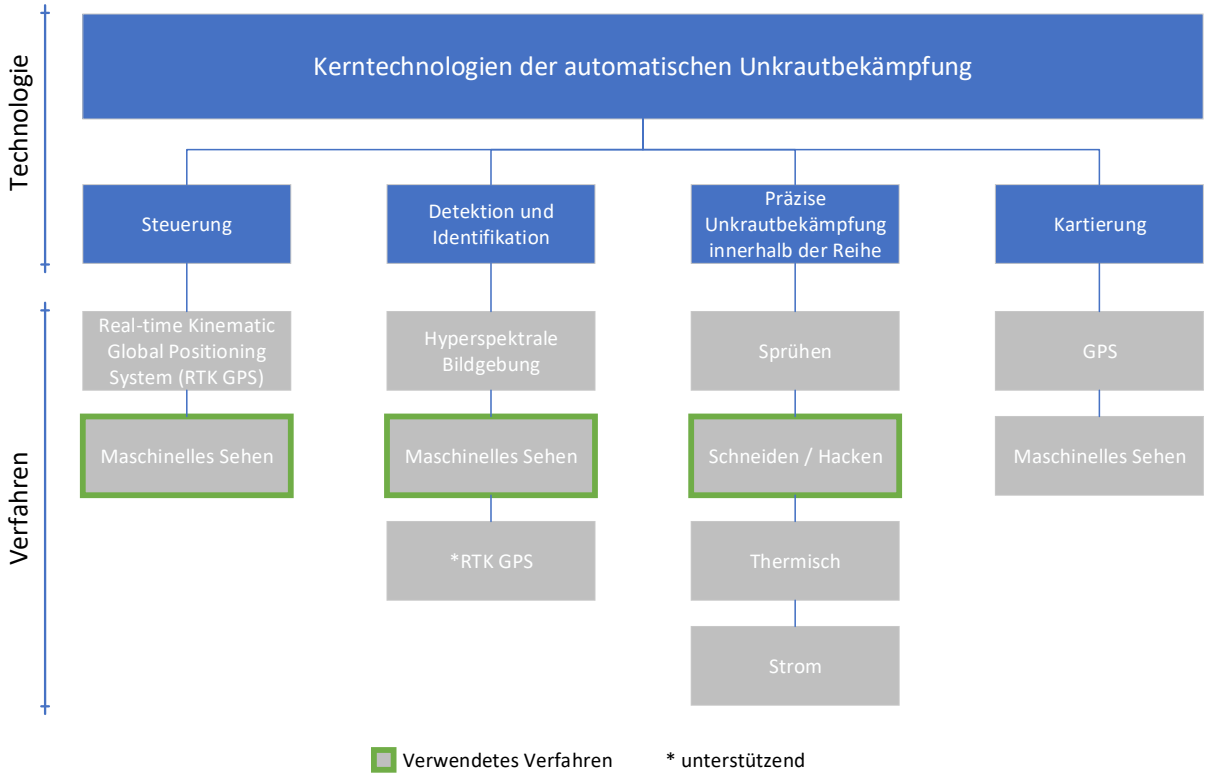


Abbildung 1-1: Kerntechnologien der automatischen Unkrautbekämpfung

Dazu zählen neben der Reihenführung, der Detektion und der Identifikation auch die präzise Unkrautbekämpfung innerhalb der Reihe und die Kartierung. **Abbildung 1-1** gibt einen Überblick darüber, welche Verfahren in der vorliegenden Arbeit verwendet werden.

Einige dieser Verfahren arbeiten mit einem Kamerasystem, um beispielsweise Pflanzenreihen auf dem Feld zu erkennen. Ein Vorteil solcher Systeme ist die vergleichsweise hohe Genauigkeit gegenüber dem in der Landwirtschaft häufig eingesetzten „global positioning system“ (GPS).

Das Funktionsprinzip von GPS beruht auf der Laufzeitbestimmung von Signalen zwischen Satelliten und Empfängern. GPS wurde ursprünglich für den militärischen Bereich entwickelt. Seit der Abschaltung der künstlichen Signalverschlechterung im Jahr 2000 ermöglicht es jedoch auch im zivilen Bereich eine Genauigkeit von 1.0 m bis 5.0 m [3]. Viele Fruchtsorten werden jedoch mit einem Reihenabstand von 0.5 m angepflanzt, was die mechanische Unkrautbekämpfung mit herkömmlichen GPS daher nicht gestattet. Um die Genauigkeit von GPS zu steigern, wird das sogenannte „differential global positioning system“ (DGPS) eingesetzt. Dabei handelt es sich um ein System, das zumindest über zwei GPS-Stationen verfügt. Eine Station ist dabei ortsfest, die andere nicht. Die ortsfeste GPS-Station dient der ortsveränderlichen als Referenzpunkt. Beiden Empfängern sind dabei die Bahnen und Uhrenfehler der Satelliten nicht genau bekannt, jedoch sind die Einflüsse für beide gleich. Das DGPS nutzt das aus, um mittels geeigneter Auswertungsmethoden die Fehler zu eliminieren [4, p. 117]. Ist der Empfänger durch Bäume oder Gebäude verdeckt, kann es jedoch auch bei DGPS zu Aussetzern kommen. Hier kommt die sogenannte Koppelnavigation (engl. „dead reckoning“) zum Einsatz. Steht das GPS-Signal aus irgendwelchen Gründen nicht zur Verfügung, wird die Position des Fahrzeugs aufgrund seiner Bewegungsrichtung und der Geschwindigkeit bestimmt. Hierbei handelt es sich im Vergleich zum GPS um eine relative Positionierung. GPS-Empfänger, welche die Methode des DGPS mit der Koppelnavigation vereinen, werden als „Real Time Kinematic differential global positioning system“ (RTK DGPS) Empfänger bezeichnet. Die Genauigkeit solcher Empfänger liegt bei wenigen Zentimetern [5]. S. Han et al. [6] weisen darauf hin, dass für chemische Anwendungen in der Landwirtschaft eine Genauigkeit von 0.04 m bis 0.12 m erforderlich ist, um Aussetzer und Überlappungen zu vermeiden. Für Anwendungen wie den Pflanzenanbau und die Ernte gibt dieses Autorenteam sogar nur einen maximalen Fehler von 0.02 m bis 0.04 m an. Zwar wären derartige Genauigkeiten beispielsweise mit DGPS-Geräten, welche zusätzlich die Phasenverschiebung der Trägerwelle auswerten, durchaus erreichbar, doch schrecken die hohen Kosten in der Regel ab [6].

Im Gegensatz dazu halten Kamerasysteme auch in modernen Fahrzeugen immer mehr Einzug. Das hat zur Folge, dass die Kosten für solche Systeme vergleichsweise gering sind. Sogenannte ADAS („Advanced Driver Assistance Systems“) werden unter anderem zur Fußgängererkennung, Fahrzeugerkennung, Kollisionsvermeidung oder auch als Spurhalteassistent eingesetzt. Um derartige Systeme auf den Markt zu bringen, benötigt es neben der Entwicklungszeit auch eine gewisse Zeit, um die konzipierten Methoden ausreichend zu testen. Aufgrund der kurzen Zeit, in der auf dem Feld getestet werden kann, ist es oft hilfreich, das Kalibrieren und funktionale Testen entwickelter Systeme ins Labor zu verlagern. Hierfür bietet sich ein „Software in the Loop“ (SiL) Prüfstand an. Mit dessen Hilfe kann die Zuverlässigkeit von Algorithmen, zum Beispiel zur Erkennung von Pflanzenreihen, evaluiert werden. Diese Evaluierung ist aufgrund der

Verfügbarkeit der „Ground Truth“ Daten in der Simulation deutlich einfacher als bei Feldversuchen [7]. Ein SiL-Prüfstand kann sowohl in der Entwurfsphase als auch zur Prüfung des fertigen Systems eingesetzt werden [8]. Wird eine virtuelle Umgebung in das SiL-Konzept eingebunden, so ist es möglich, reproduzierbare Tests mit unterschiedlichen Witterungsverhältnissen, Kamerasettings, Ackerflächen und Pflanzenarten ganzjährig durchzuführen.

> Das Ziel der vorliegenden Arbeit ist es daher, einen „Software in the Loop“ Prüfstand zu entwickeln, mit dessen Hilfe ein bestehender Algorithmus zur Pflanzenreihen-Erkennung getestet und evaluiert werden kann.

Entwickelt wurde dieser Algorithmus in der Master Thesis „Vision-Based Crop Row Detection and Obstacle Recognition for Precision Farming“ [9], die im Zuge des Masterstudiums „Mechanical Engineering“ an der FH Technikum Wien erstellt wurde.

Auf die Notwendigkeit der automatischen Unkrautregulierung weisen D.C. Slaughter et al. in [2] hin. Sie erklären in ihrer Arbeit, dass Unkraut mit Kulturpflanzen um Feuchtigkeit, Nährstoffe und Sonnenlicht konkurriert. Unter anderem verweisen sie auf [12], der die Ertragsreduktion von direkt gesäten Tomaten bei 71% sieht, wenn zusätzlich der Stechapfel („Datura stramonium L.“) auf dem Feld wächst. Jeweils unter der Annahme, dass 11 Unkrautpflanzen pro Quadratmeter Feld wachsen, liegt die Ertragsreduktion bei großen Sternwinden („Ipomoea L.“) bei 67% und bei der Gewöhnlichen Spitzklette („Xanthium strumarium L.“) sowie der Blutroten Fingerhirse („Digitaria sanguinalis (L.) Scop.“) immerhin noch bei 48%.

Als größte Herausforderung sehen D.C. Slaughter et al. [2] die Detektion und Identifikation von Unkraut unter den vielfältigen Bedingungen, die auf den Feldern herrschen. Im Rahmen dieser Arbeit soll somit untersucht werden, ob es mittels Simulation möglich ist, derartig vielfältige Bedingungen nachzubilden. Etliche wissenschaftliche Arbeiten der letzten Jahre nutzen synthetische Bilder für diverse Simulationen; dennoch findet man bis heute nur eine überschaubare Zahl an Publikationen, die sich mit dem Thema „Evaluierung von bildgestützten Algorithmen mithilfe von synthetischen Bildern“ auseinandersetzen. Die vorliegende Arbeit behandelt das Thema daher umso kritischer, um schlussendlich festzustellen, dass es möglich ist, bildgestützte Algorithmen mithilfe synthetischer Bilder zu testen.

2 VERWANDTE ARBEITEN

J.M. Chandler und F.T. Cooke [11] erläutern in ihrer Publikation „Economics of cotton losses caused by weeds“, die 1992 erschienen ist, die hohen Kosten und die vergleichsweise geringe Präzision des Handhackens. Womöglich war es einer dieser beiden Faktoren, der in den letzten Jahren etliche Wissenschaftler dazu bewog, sich mit dem Thema automatische Unkrautregulierung auseinanderzusetzen. Egal welchen Ansatz sie dabei gewählt haben, sei es mittels GPS [12], mithilfe einer Stereokamera [13], unter Zuhilfenahme eines Infrarotfilters [14] oder wie in der vorliegenden Arbeit mit einer Farbkamera: Eine ausreichende Evaluierung des Systems war auf jeden Fall vonnöten.

Die Idee, Spiele Engines für wissenschaftliche Zwecke zu verwenden, ist nicht neu. M. Cicco et al. [15] erstellten mithilfe der „Unreal Engine“ [16] ein virtuelles Zuckerrübenfeld, mit dessen Hilfe ein neuronales Netzwerk zur Unkrauterkennung trainiert wird. Mancini et al. [17] nutzten die „Unreal Engine“, um ein städtisches Szenario abzubilden, und Tschentscher et al. [18] entwickelten in der „Unreal Engine“ eine virtuelle Kamera zur Parkraumüberwachung. Die Möglichkeiten, die moderne Spiele Engines bieten, sind nahezu unbegrenzt. T. Theuerkauff et al. [19] banden ein digitales Oberflächenmodell des Wasserstraßen- und Schifffahrtsamts Bremerhaven, das mittels Fächerecholot erzeugt wurde, in eine virtuelle Umgebung ein, um das Gelände unter Wasser nachzustellen. Dabei erlaubt ihnen das metrische „Unreal-Koordinatenreferenzsystem“ eine direkte Übernahme des Geländemodells ohne zusätzliche Koordinatentransformation.

Y. Zhang et al. [20] führten das große Interesse an Computergrafik darauf zurück, dass es verhältnismäßig einfach ist, eine große Zahl an Bildern mit „Ground Truth“ Daten zu generieren. Synthetische Daten wurden bereits in der Stereoanalyse, der Berechnung des optischen Flusses, der Detektion und der semantischen Segmentierung verwendet [20].

Wie einleitend bereits erwähnt, finden sich in der Literatur dutzende Publikationen, die Teilgebiete dieser Arbeit abdecken. Eine Veröffentlichung, die bildgestützte Algorithmen für die Landwirtschaft mittels synthetischer Bilder evaluiert, sucht man jedoch bislang vergeblich.

3 ALTERNATIVE LÖSUNGSWEGE

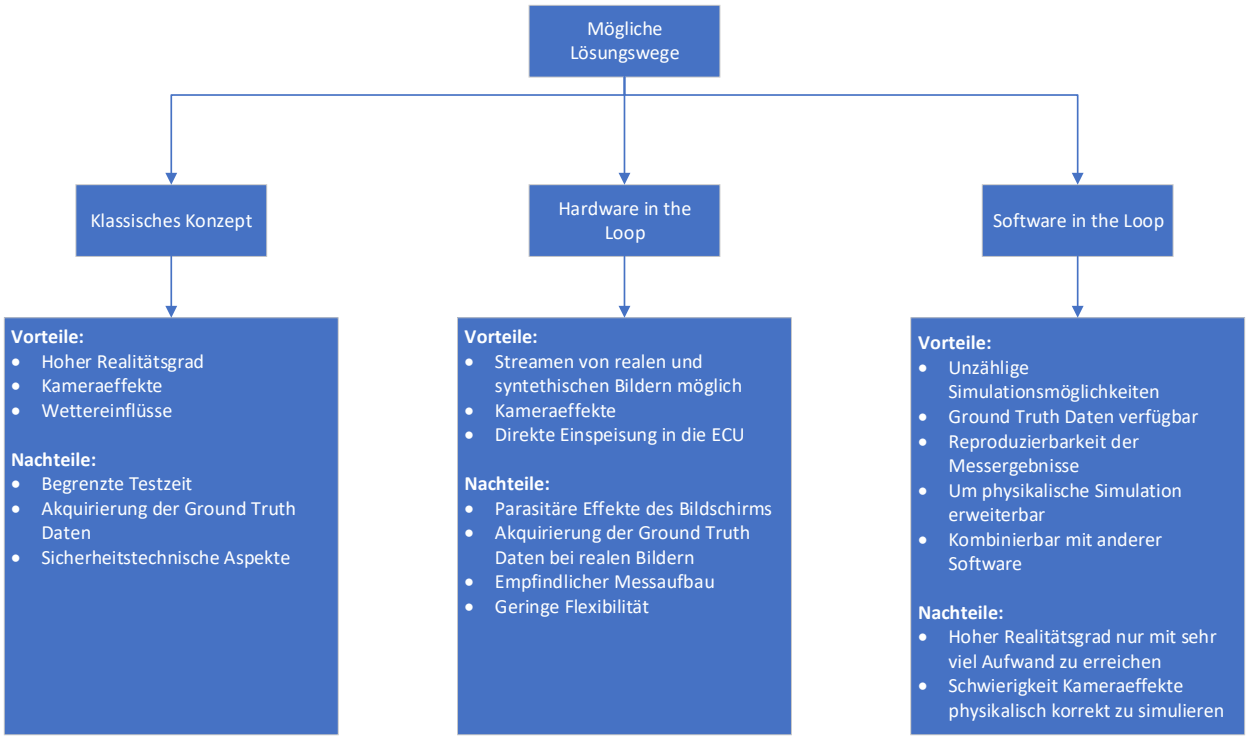


Abbildung 3-1: Vor- und Nachteile alternativer Lösungswege

Abbildung 3-1 soll die Vor- und Nachteile unterschiedlicher Konzepte zur Evaluierung von bildgestützten Algorithmen aufzeigen. Vorgestellt werden drei unterschiedliche Konzepte: Das klassische Konzept, bei dem der entwickelte Algorithmus unter realen Bedingungen direkt am Feld getestet wird, das „Hardware in the Loop“ (HiL) Konzept, bei dem eine Kamera von einem Bildschirm stimuliert wird, und das „Software in the Loop“ (SiL) Konzept, bei welchem die Kamera mittels Software simuliert wird.

Das wohl bekannteste Konzept ist das „klassische Testen“ auf dem Feld. Die Vorteile liegen auf der Hand. Eine Simulation setzt immer das Vorhandensein eines Modells voraus, das wiederum eine vereinfachte Abbildung der Wirklichkeit ist. Grundsätzlich gilt, dass sich die Vielfältigkeit der Natur nicht in einer Simulation abbilden lässt. Verschiedenste Sonnenstände und die unterschiedlichsten Lichtverhältnisse, die mit dem wechselnden Wetter einhergehen, lassen das Grün der Pflanzen immer anders erscheinen. Beim Testen auf dem Feld erreicht man daher stets das Maximum an Realität. Parasitäre Effekte wie Bilderschütterungen, Verschmutzungen an der Linse oder andere Kameraeffekte werden direkt in das Steuergerät (engl. „electronic control unit“, ECU) eingespeist und zeigen unmittelbar deren Auswirkungen auf den Algorithmus. Neben potenziellen sicherheitstechnischen Aspekten ist die aufwendige Akquirierung der „Ground Truth“ Daten der wohl größte Nachteil dieses Konzepts. Beispielsweise steckten H. Behfar et al. [21] zwei Stangen in den Boden, spannten ein Seil dazwischen und setzten Linsensamen in die

Furche unterhalb des Seils, um eine 40 m lange Teststrecke mit bekannten Pflanzenpositionen zu erhalten. Ein weiterer nicht unwesentlicher Nachteil ist die kurze Zeit, in der auf dem Feld getestet werden kann. Je nach Fruchtsorte liegt die Zeitspanne für die Unkrautbekämpfung bei wenigen Wochen. Zwar wird der Einfluss des Wetters in **Abbildung 3-1** als Vorteil für das klassische Konzept angeführt, doch kann es vorkommen, dass aufgrund der kurzen Zeitspanne nicht sämtliche Szenarien mit unterschiedlichsten Witterungsverhältnissen getestet werden können. Es ist nicht davon auszugehen, dass man in einem absehbaren Zeitraum auf reale Tests vollständig verzichten kann. SiL- und HiL-Konzepte ermöglichen jedoch bereits heute die Simulation während der Entwicklungsphase; sie können dazu beitragen, zeitaufwendige und oft teure reale Tests zu reduzieren.

Beim „Hardware in the Loop“ Konzept wird die Kamera auf einen Bildschirm gerichtet, der aufgenommene Bilder von Realfahrten oder simulierte Bilder zeigt. Auf den ersten Blick erscheint es fast so, als würde das HiL-Konzept zwei wesentliche Vorteile des klassischen Konzepts übernehmen. Einerseits lassen sich damit Kameraeffekte simulieren und andererseits kann man aufgenommene reale Videos beliebig oft abspielen, um Anpassungen am Algorithmus vorzunehmen. Die Einspeisung in das Steuergerät erfolgt, wie auch schon beim klassischen Konzept, direkt aus der Kamera. M. Haselhoff und S. Hakuli [22] zeigen allerdings signifikante Schwachstellen dieses Konzepts auf. So kann eine fehlende Synchronisation zwischen dem Bildaufbau im Monitor und der Bilderfassung in der Kamera „zerrissene“ Bilder zur Folge haben. Weiters sehen die Autoren einen Nachteil in der Reaktionszeit einzelner Zellen. Die Helligkeit der einzelnen Pixel wird durch Verdrehen der Flüssigkristalle gegenüber einem Polarisationsfilter gesteuert, wodurch mehr oder weniger Licht der Hintergrundbeleuchtung zur Bildschirmoberfläche durchgelassen wird. Die Zeitspanne für diesen Vorgang entspricht in etwa der üblichen Belichtungszeit der Kamera, was zur Überlagerung des aktuellen Bildinhalts mit dem vorhergehenden führen kann. Für einen Algorithmus, der die Position aufgrund des Bildinhalts bestimmt, kann eine derartige Überlagerung bereits zu Problemen führen. Der vorliegende Algorithmus verwendet Graustufenbilder (siehe Abschnitt 4.2), um Pflanzen von ihrem Hintergrund zu unterscheiden. Je nach den Lichtverhältnissen muss er dafür in der Lage sein, bereits geringe Helligkeitsunterschiede zu detektieren. M. Haselhoff und S. Hakuli [22] stellten fest, dass der Kontrast bei Monitoren niedriger ist als bei modernen Bildsensoren, was dazu führt, dass Blendeffekte nicht realistisch darstellbar sind. Als weiteren Nachteil führen die Autoren die Komplikationen an, die auftreten, wenn Kameras mit einem „Field of View“ (FOV) von mehr als 180° getestet werden sollen. Herkömmliche ebene Bildschirme sind für Szenendarstellungen mit derartigen Kameras, die unter anderem bei Einparkassistenten eingesetzt werden, ungeeignet. Zur Stimulation einer Stereokamera müssen laut M. Haselhoff und S. Hakuli [22] zwei „Videostreams“ mittels Vorsatzlinsen, Spiegel oder Polarisationsfilter in die ECU eingespeist werden. Für alle genannten Methoden bedarf es eines sehr präzisen Aufbaus, was das „Hardware in the Loop“ Konzept im Allgemeinen sehr störanfällig macht.

Einen deutlich robusteren Aufbau verspricht das „Software in the Loop“ Konzept. Zumindest die vorliegende Arbeit verzichtet bei ihrem Aufbau vollkommen auf Hardware-Komponenten. Das hat zur Folge, dass äußere Störeinflüsse keinen Einfluss auf die Ergebnisse haben. Wie auch schon beim HiL-Konzept, erlaubt das „Software in the Loop“ Konzept das Abspielen von realen und synthetischen Videos. Der Vorteil des Abspielens realer Videos liegt ganz klar im maximierten Realismus. Ein wesentlicher Nachteil ist jedoch die fehlende Möglichkeit einer Rückkoppelung der Lenkkorrektur (vgl. **Abbildung 4-1**). Die vorliegende Arbeit setzt daher auf eine virtuelle Umgebung, in der eine Kamera, deren Position zur Laufzeit verändert werden kann, simuliert wird. Erstellt wird diese virtuelle Umgebung in der „Unreal Engine“ (UE) [16]. Die Vorteile dieses Konzepts sind vielfältig. Eine große Benutzergemeinschaft der „Unreal Engine“, die unter anderem aus Künstlern und Entwicklern besteht, entwirft visuell realistische Umgebungen und Objekte, die man einem beliebigen Projekt ohne großen Aufwand hinzufügen kann [23]. Dieses Vorgehen ermöglicht es, unzählige Simulationsumgebungen zu schaffen. Neben visuell realistischen Umgebungen lassen sich auch physikalische Simulationen mit der UE durchführen, was den Rahmen der Möglichkeiten nochmals erweitert. Die C++ Programmierschnittstelle (engl. „application programming interface“, API) gestattet die bidirektionale Kommunikation der virtuellen Umgebung mit anderer Software. Somit lassen sich beispielsweise Berechnungen mit „Matlab“ durchführen oder Simulationsergebnisse in ein Textfile schreiben. Diese Schnittstelle ermöglicht auch den Zugang zu einem der wohl größten Vorteile dieses Konzepts. Mithilfe des SiL-Konzepts ist es möglich, jederzeit sämtliche die Simulation betreffenden Daten auszulesen und zu bewerten. Dazu zählen unter anderem die „Ground Truth“ Daten, Daten über die Licht- und Wetterverhältnisse, geometrische Eigenschaften der verschiedensten Objekte, allgemeine Daten, welche die Simulation betreffen wie Bildauflösung, Übertragungsraten, Kameraparameter, und noch viele mehr. Diese Daten sind notwendig, um reproduzierbare Tests zu ermöglichen. Für die Verwendung des Konzepts in der wissenschaftlichen Praxis sind diese reproduzierbaren Tests unabdingbar. Als Nachteile gegenüber den beiden anderen Konzepten sind der hohe Aufwand, der mit steigendem Realismus einhergeht, und die virtuelle Kamera zu nennen. Beide Punkte lassen sich trotz interdisziplinärer Zusammenarbeit von Künstlern und Entwicklern nicht ganz kompensieren. Zweifellos werden zukünftige Spiele Engines die Leistungen der heutigen noch weit übertreffen, der Realitätsgrad der Natur wird dabei jedoch nie erreichbar sein.

Trotz des vergleichsweisen hohen Aufwands gegenüber dem klassischen Konzept setzt die vorliegende Arbeit auf das SiL-Konzept. Die oben genannten Vorteile überwiegen die Nachteile deutlich. Das HiL-Konzept bietet, sofern es ebenfalls über eine virtuelle Umgebung stimuliert wird, zwar ähnliche Vorteile wie das SiL-Konzept; die Nachteile, die das Filmen eines Bildschirms mit sich bringt, sind allerdings zu groß.

4 DER PRÜFSTAND

In der Automobilindustrie werden gerade bei sicherheitsorientierten Entwicklungen von Fahrerassistenzsystemen (FAS) funktionale Anforderungen bereits in der Modellphase mittels „Model in the Loop“ (MiL), „Software in the Loop“ (SiL) oder „Hardware in the Loop“ (HiL) abgesichert. Die modellbasierte Entwicklung bietet den Vorteil, dass immer mehr Entwicklungsschritte an den Anfang der Entwicklungsphase verlegt werden können [24]. So können etwaige Fehler früher erkannt und Kosten gespart werden. Weitere Vorteile sind nach J. Meyer [24] die Möglichkeit, die Reaktion von Funktionen auf fehlerhafte Sensordaten bzw. Sensorausfälle zu testen und damit das Fail-Safe-Verhalten abzusichern, sowie die Entstehung einer Testbibliothek während der Entwicklungsphase.

Bei dem in dieser Arbeit entwickelten Prüfstand handelt es sich um einen „Software in the Loop“ Prüfstand, der die oben genannten Vorteile für die Entwicklung von Algorithmen umsetzen kann und zusätzlich sämtliche Anforderungen erfüllt, die für die wissenschaftliche Forschung erforderlich sind.

Yi Zhang et al. [20] beschreiben die Standardstrategie der wissenschaftlichen Forschung wie folgt:

„changes variables separately and systematically and study their impact“.

Dieser Grundsatz wurde bei der Entwicklung des Prüfstandes stets berücksichtigt. Neben unzähligen Variablen, welche die virtuelle Umgebung betreffen, lassen sich auch die Übertragungsrate, die Auflösung der synthetischen Bilder, die Bewertungsmethode und vieles mehr anpassen. Damit soll erreicht werden, dass der Prüfstand zukünftig individuell einsetzbar ist. Mögliche Anwendungsfälle sind neben der Weiterentwicklung der Algorithmen, um das landwirtschaftliche Fahrzeug vollständig autonom auf dem Feld fahren zu lassen, das Testen alternativer Kamerasysteme.

4.1 Funktionsprinzip

Abbildung 4-1 zeigt den grundsätzlichen Aufbau des „Software in the Loop“ Prüfstands. Dieser basiert auf vier Komponenten: der virtuellen Umgebung, erstellt im Zuge dieser Arbeit in der „Unreal Engine 4“, dem Algorithmus für die Pflanzenreihenerkennung, programmiert in der Diplomarbeit „Vision-Based Crop Row Detection and Obstacle Recognition for Precision Farming“ [9], einem „Matlab-Skript“, das die Messergebnisse auswertet, und einem Datenspeicher, um Daten dauerhaft abzuspeichern. Die bidirektionale Kommunikation zwischen virtueller Umgebung (VR) und Algorithmus wird mithilfe des „Robot Operation Systems“ (ROS) bewerkstelligt (siehe Abschnitt 4.3).

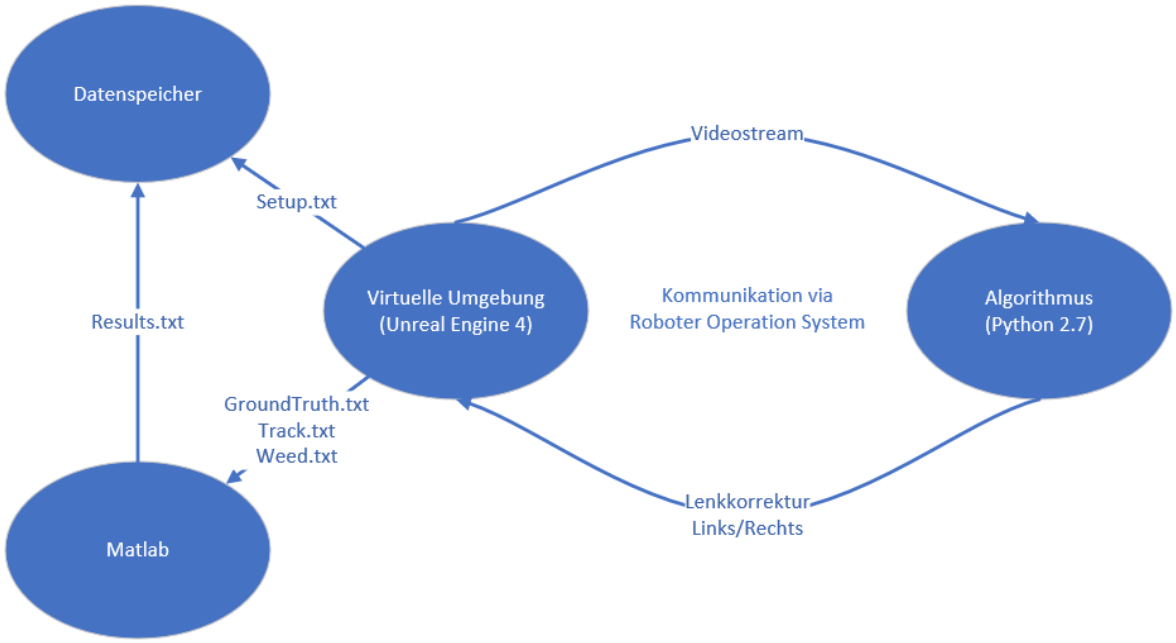


Abbildung 4-1: Funktionsprinzip „Prüfstand“

Die VR stellt dem Algorithmus einen „Videostream“ mit konstanter Bildwiederholungsfrequenz zur Verfügung. Dieser ermittelt den „Cross Track Error“ (vgl. [9, Ch. 4.6]) und sendet ihn zurück an die virtuelle Umgebung. Diese berechnet anschließend mithilfe des CTE Lenkkorrekturen und steuert somit das Zugfahrzeug. Die Messdaten werden mittels „Matlab“ ausgewertet und anschließend im Datenspeicher mit einem Zeitstempel abgelegt. Diese systematische Speicherung aller relevanten Daten macht es möglich, reproduzierbare Messergebnisse zu generieren.

4.2 Der Algorithmus

Um die Evaluierungsmethoden der vorliegenden Arbeit besser nachvollziehen zu können, stellt dieser Abschnitt den Algorithmus in aller Kürze vor. Entwickelt wurde der Algorithmus von R. Deutsch [9] in seiner Master Thesis „Vision-Based Crop Row Detection and Obstacle Recognition for Precision Farming“. Der Quellcode wurde in Python 2.7 geschrieben und ist daher kompatibel mit dem „Robot Operating System“. Die grundsätzliche Funktionsweise ist in **Abbildung 4-2** mithilfe eines Flussdiagramms beschrieben.

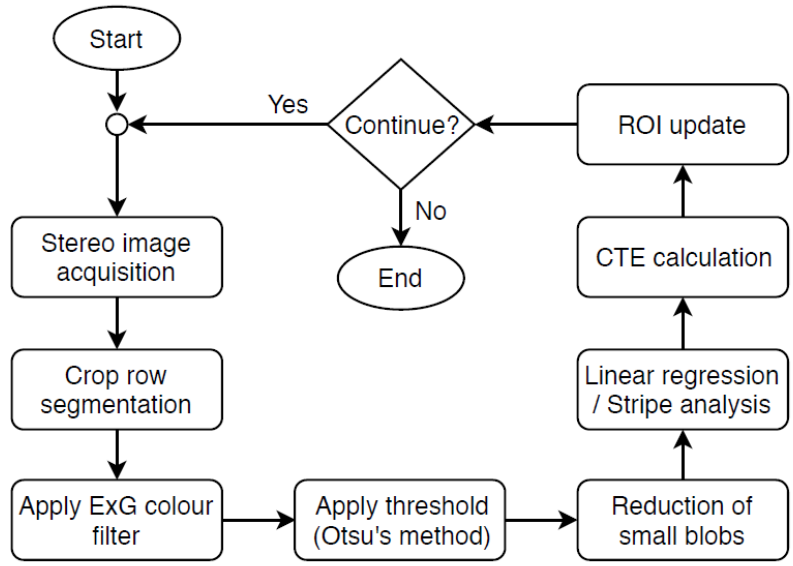


Abbildung 4-2: Flussdiagramm des Algorithmus zur Reihendetektion

Quelle: R. Deutsch [9, Fig. 14]

Etliche Algorithmen zur Reihendetektion basieren auf drei wesentlichen Schritten: der Bilderfassung, der Bildvorverarbeitung und dem „Line Fitting“. Auch der Algorithmus von R. Deutsch [9] beinhaltet diese Schritte, allerdings werden diese durch vor- und nachgelagerte Schritte erweitert. R. Deutsch entwickelt in seiner Arbeit neben dem Algorithmus zur Reihendetektion auch einen Algorithmus zur Hinderniserkennung. Dieser basiert auf der Analyse von Stereo-Bildpaaren (vgl. [9, Ch. 5]), weshalb in **Abbildung 4-2** die „Stereo image acquisition“ als erster Schritt angeführt wird. Für die Reihendetektion wird hingegen zu Beginn ein Farbbild aufgenommen. Das erfasste Bild enthält etliche Information, die für die Reihendetektion nicht von Relevanz sind. Um Rechenzeit zur sparen, werden diese Informationen daher nicht berücksichtigt und nur jener Bereich untersucht, der tatsächlich von Interesse ist. In weiterer Folge ist dieser Bereich mit „Region of Interest“ (ROI) bezeichnet. Anschließend wird innerhalb dieser ROI mithilfe des „Excess green index“ (ExG) versucht, Pflanzen vom Boden zu separieren. Das Ergebnis ist ein Graustufenbild, das mittels Otsus Schwellwertmethode (vgl. [25]) zu einem Binärbild transformiert wird. Weiße Pixel sollen dabei Pflanzen darstellen, schwarze Pixel den Boden. Dieses Binärbild enthält neben Pflanzen auch Unkraut, das im Bild durch weniger zusammenhängende weiße Pixel dargestellt ist. Diese weißen Pixel würden sich negativ auf das „Line fitting“ (siehe Abschnitt 4.5) auswirken, weshalb sie aus dem Bild entfernt werden. Sind nur mehr die relevanten Pixel im Bild vorhanden, wird mithilfe der „Linearen Regression“ bzw. der „Stripe Analysis“ versucht, die Pflanzenreihe mittels einer Geraden zu approximieren. Diese Gerade dient dazu, den „Cross Track Error“ (CTE) zu berechnen. Der CTE gib Auskunft darüber, wie weit sich das Zugfahrzeug von der Ideallinie entfernt hat. Zu guter Letzt wird die „Region of Interest“ neu angepasst und der komplette Vorgang beginnt erneut.

4.3 „Robot Operating System“

Das „Robot Operating System“ ist ein „open source Framework“ für die Entwicklung von Roboter-Software. Es beinhaltet Werkzeuge, Bibliotheken und Konventionen mit dem Ziel, die Entwicklung von Robotern zu vereinfachen [26, p. 3]. Entstanden ist es aus unterschiedlichen Projekten der Stanford University in der Mitte der 2000er Jahre [26, p. 4]. Zur Erstellung dieser Arbeit wurde „ROS Melodic Morenia“ (das offiziell 12. Release) verwendet.

Die vorliegende Arbeit verwendet ROS, da es so möglich ist, die virtuelle Umgebung und den entwickelten Algorithmus auf unterschiedlichen Plattformen zu betreiben. Zukünftig soll das den Vorteil bringen, dass bei der Entwicklung von Algorithmen nicht zusätzlich noch die speicher- und rechenintensive virtuelle Umgebung auf dem Notebook installiert werden muss.

4.3.1 Aufbau

Um den grundsätzlichen Aufbau des vorliegenden Frameworks zu verstehen, werden zuerst die Kernelemente des ROS beschrieben. Diese sind „nodes“, „topics“ und „messages“.

Bei „nodes“ handelt es sich um ausführbare Prozesse, die sämtliche Berechnungen eines ROS Frameworks übernehmen. Sie werden durch C++ oder Python Quellcode beschrieben und sind in der Lage, miteinander zu kommunizieren. Diese Kommunikation geschieht in den meisten Fällen über sogenannte „topics“. Das „topic“ ist dabei jedoch nur die Bezeichnung des Datenflusses. Der Datenfluss wiederum besteht aus „Messages“ mit definiertem Datentyp [26, p. 31]. Jeder „node“ ist fähig, für ihn relevante „topics“ zu abonnieren (engl. „subscribe“) bzw. Rechenergebnisse zu veröffentlichen (engl. „publish“). **Abbildung 4-3** zeigt das dieser Arbeit zugrunde liegende ROS Framework. „Nodes“ sind hier durch Ellipsen und „topics“ durch Rechtecke dargestellt. Die wichtigsten „nodes“ und „topics“ werden im Folgenden kurz vorgestellt.

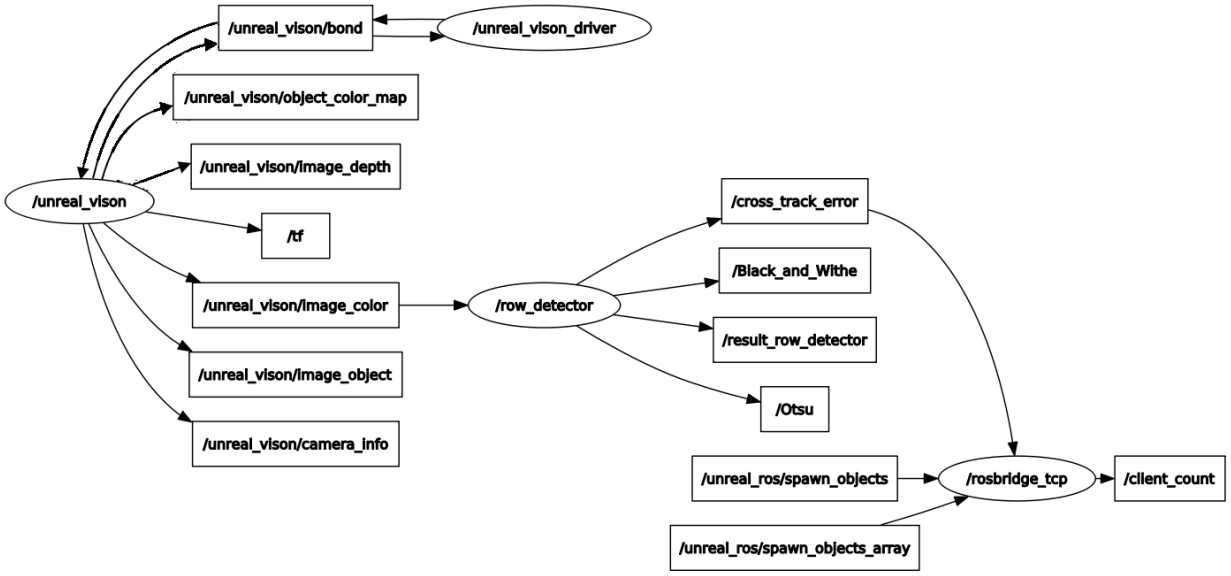


Abbildung 4-3: ROS Framework SiL-Prüfstand

Beim „node unreal_vision“ handelt es sich um ein Plugin, das im August 2018 auf der GitHub Homepage veröffentlich wurde. Es ermöglicht die Kommunikation zwischen der „Unreal Engine“ und dem bestehenden Algorithmus mittels „Videostream“ [27]. Neben dem in dieser Arbeit verwendeten Farbbild, das im „topic“ „unreal_vision/Image_color“ übertragen wird, werden noch das Tiefenbild, sowie diverse Marker für die Visualisierung und Informationen zur Kamera zur Verfügung gestellt.

Der „node” „row_detector“ wurde in der Diplomarbeit „Vision-Based Crop Row Detection and Obstacle Recognition for Precision Farming“ erstellt. Er enthält den Quellcode für den Algorithmus. Dieser abonniert das vom „node“ „unreal_vision“ veröffentlichte „topic“ „unreal_vision/Image_color“, verarbeitet diese Information und veröffentlicht die Ergebnisse. Neben den Resultaten der Pflanzenreihendetektion und des CTE werden auch das Schwarzweiß-Bild und das Bild nach der Otsu Schwellwertmethode für weitere Analysen veröffentlicht.

Ähnlich wie der „node“ „unreal_vision“ ist der „node“ „rosbridge_tcp“ auch für die Kommunikation zwischen der „Unreal Engine“ und dem bestehenden Algorithmus verantwortlich, nur werden hier anstatt des „Videostreams“ Informationen über den aktuellen „Cross Track Error“ ausgetauscht.

4.4 Erzeugte Daten

Um reproduzierbare Messergebnisse bereitstellen zu können, ist es notwendig, nicht nur die Messergebnisse abzuspeichern, sondern auch detaillierte Informationen zum Messaufbau zu sichern. Diese Informationen beinhalten das Setup sowie die „Ground Truth“ Daten. Das Setup enthält alle relevanten Einstellungen, wie sie in der „Unreal Engine“ (UE) für diese Prüfung

getroffen wurden. **Tabelle 4-1** stellt einen Überblick über die für diese Messreihe relevanten Daten dar. Die „Unreal Engine“ bietet unzählige Einstellungsmöglichkeiten, weshalb diese Daten von Messreihe zu Messreihe variieren können.

Geometry	Post Processing Setup	SunSky Plugin
X_Distance between the crops [cm]	Metering Mode	Sky Light Intensity
Y_Distance between the crops [cm]	Dirt Mask Texture	Sky Light Color (BGR)
Min_Offset [cm]	Dirt Mask Intensity	Directional Light Intensity [Lux]
Max_Offset [cm]	Shutter Speed [1/s]	Directional Light Temperature [K]
Min_Scale	ISO	Directional Light Color (BGR)
Max_Scale	Arperture (F-stop)	Longitude
Min_Rotation [degree]	Grain Jitter	Latitude
Max_Rotation [degree]	Grain Intensity	North Offset [degree]
Plants per row	Near Blur Size [cm]	Time Zone
Weed density [#/m²]	Far Blur Size [cm]	Month
Camera hight [cm]	Motion_Blur Amount	Day
Camera pitch angle [degree]	Motion Blur Max	Solar Time
Tractor speed [m/s]	Target FPS	Fog Density
		Fog Height Falloff

Tabelle 4-1: Auszug aus der Datei „Setup.txt“

Die Datei „Ground_Truth.txt“ beinhaltet Information über die Position der Pflanzen, deren Rotation und deren Skalierung. Die Kenntnis der „Ground Truth“ Daten ist mitunter einer der größten Vorteile der Evaluierung von Algorithmen mit synthetischen Bildern. Mithilfe dieser Daten ist es möglich, einen idealen Pfad für das Implement zu berechnen und diesen mit der tatsächlich gefahrenen Linie aus der Datei „Track.txt“ zu vergleichen (siehe Abschnitt 4.6). Die Ergebnisse dieses Vergleichs werden abschließend in der Datei „Results.txt“ abgelegt (siehe **Tabelle 4-2**).

Neben den eigentlichen Messergebnissen wird noch die Funktionsfähigkeit des Prüfstandes evaluiert. Es wird die Frequenz der eingehenden Lenkkorrekturen überwacht; sollte es hier zu größeren Abweichungen kommen, so kann die Messung für ungültig erklärt werden. Zusätzlich werden die Messergebnisse graphisch ausgewertet. Die „Unreal Engine“ ist aktuell nur in Englisch verfügbar [28], weshalb die Einstellungen in den vorliegenden Auswertungen stets mit den originalen englischen Begriffen bezeichnet sind. Der Vollständigkeit halber sei noch die Datei „Weed.txt“ erwähnt, die ähnlich wie die Datei „Ground_Truth.txt“ die Positionen, die Rotationen und die Skalierungen des platzierten Unkrauts enthält.

Measurment results	Status test stand
Data basis	Start offset
Racing line Standard deviation	Steering command Standard deviation
Racing line Maximum deviation	Steering command Maximum deviation
Mean deviation Racing line	Steering command Mean
Mean deviation Racing line (absolut values)	
CTE Standard deviation	
CTE Maximum deviation	
Mean deviation Racing line	
Mean deviation Racing line (absolut values)	

Tabelle 4-2: Auszug aus der Datei „Results.txt“

4.5 Approximationsverfahren

Nach Abbildung 4-1 ist die Reihendetektion mittels Approximationsverfahren der 6. Schritt im Algorithmus. Diesem Schritt gebührt besondere Aufmerksamkeit, da die Pflanzenreihe durch eine Gerade approximiert wird und diese für den Algorithmus somit keine willkürliche Verteilung an Pixeln mehr darstellt. Beim sogenannten „Line fitting“ wird versucht, eine Linie zu finden, die am besten zu den weißen Pixeln im Binärbild passt. Eine der am meisten verwendeten Verfahren zur Erkennung von Pflanzenreihen ist die Hough Transformation. Sie eignet sich hervorragend, wenn aufgrund von unvollständiger Keimung, Insektenschäden oder anderen Faktoren Lücken in den Reihen auftreten [2]. V. Fontaine und T.G. Crowe [29] analysieren mehrere Approximationsverfahren und kommen zu dem Resultat, dass die „Lineare Regression“ bei Aufnahmen, die auf dem Feld gemacht wurden, der „Hough Transformation“ gleichwertig ist, allerdings liefert sie bei Bildern mit viel Unkraut ein etwas besseres Ergebnis. Der in dieser Arbeit eingesetzte Algorithmus lässt die Wahl zwischen „Linearer Regression“ und der „Stripe Analysis“. Die Abschnitte 4.5.1 und 4.5.2 gehen daher auf diese beiden Verfahren näher ein.

4.5.1 „Lineare Regression“

Als Pionier der regressionsanalytischen Untersuchungen gilt der Brite Sir Francis Galton (1822–1911), der sich gegen Ende des 19. Jahrhunderts mit der Frage der Vererbung beschäftigte. Mehr als 100 Jahre später finden lineare Regressionsmodelle immer noch etliche Anwendungen in der Statistik [30, p. 1]. Das einfachste mögliche Regressionsmodell lautet

$$y = \beta_0 + \beta_1 x + \varepsilon \qquad 4.1$$

wobei $\beta_0 + \beta_1 x$ linear sind und die Störgröße ε die zufälligen Abweichungen zusammenfasst. Systematische Abweichungen von Null werden durch den Parameter β_0 berücksichtigt, weshalb man den Erwartungswert $E(\varepsilon_i) = 0$ annimmt. Das vorliegende lineare Regressionsmodell ist

besonders dann geeignet, wenn die Zielvariable y stetig und wenn möglich approximativ normalverteilt ist.

4.5.1.1 Lineare Regressionsmodell

Da sich in dieser Arbeit das Modell auf den eindimensionalen Fall beschränkt, soll nachfolgend das einfache lineare Regressionsmodell vorgestellt werden [29, p. 20].

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \ , i = 1, \dots, n. \tag{4.2}$$

Die Fehler $\varepsilon_1, \dots, \varepsilon_i$ sind unabhängig und identisch verteilt mit

$$E(\varepsilon_i) = 0, \ \ Var(\varepsilon_i) = \ \sigma^2. \tag{4.3}$$

Die Gerade $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$ wird als Regressionsgerade bezeichnet und kann als Schätzung $\widehat{E}(y|x)$ für den bedingten Erwartungswert y bei gegebenen Kovariablenwert x angesehene werden.

4.5.1.2 Notation

Bevor die Methode der kleinsten Fehlerquadrate (MKQ) vorgestellt wird, ist noch kurz auf die Notation einzugehen [30, p. 63]. Grundsätzlich sind geschätzte Parameter mit einem „Dach" gekennzeichnet, um sie von den wahren Parametern zu unterscheiden. Es gilt

$$\beta \neq \hat{\beta} \tag{4.4}$$

da es nicht gelingen wird, den „wahren" Parametervektor ohne Fehler zu schätzen. Geht man von den Schätzungen $\hat{\beta}_0$ und $\hat{\beta}_1$ aus, erhalten wir eine Schätzung des Erwartungswertes $E(y_i)$ von y_i durch

$$\widehat{E}(y_i) = \ \hat{\beta}_0 + \hat{\beta}_1 x_{i1} \tag{4.5}$$

Diese Schätzung bezeichnet man mit $\hat{y}_i$.

$$\hat{y}_i = \ \widehat{E}(y_i). \tag{4.6}$$

Den Schätzfehler, sprich die Abweichung des wahren Wertes y_i vom Schätzwert $\hat{y}_i$, nennt man Residuum; er wird im Folgenden mit $\hat{\varepsilon}_i$ bezeichnet. Wichtig ist es nach L. Fahrmeir et al. [30] zu verstehen, dass es sich beim Residuum nicht um die Störgröße ε_i handelt, sondern vielmehr um eine Schätzung der Störgröße.

4.5.1.3 Methode der kleinsten Fehlerquadrate

Die Regressionsparameter β_0 und β_1 werden heutzutage vorwiegend mit der Methode der kleinsten Fehlerquadrate geschätzt [30, p. 90]. Das hat zwei Gründe: Einerseits weisen die durch die MKQ gewonnenen Schätzungen eine Reihe wünschenswerter statistischer Eigenschaften auf,

andererseits ist die Schätzung der Parameter mittel MKQ mathematisch verhältnismäßig einfach, wie die folgenden Zeilen zeigen.

Aufgrund von Beobachtungen $(y_i, x_i),\ i = 1, \dots, n$ werden die Parameter so geschätzt, dass die Summe der quadratischen Abweichungen

$$\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \qquad 4.7$$

der Beobachtungen y_i von der Regressionsgeraden $\beta_0 + \beta_1 x_i$ minimal wird.

Die Schätzwerte für die unbekannten Parameter β_0 und β_1 erhalten wir demnach durch Minimierung der Summe der quadratischen Abweichungen.

$$KQ(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 = \sum_{i=1}^{n} \varepsilon_i^2 \qquad 4.8$$

Durch die partielle Ableitung nach β_0 und β_1 und anschließendem lösen des linearen Gleichungssystems, ergibt sich der Schätzer für die kleinsten Fehlerquadrate (KQ-Schätzer):

$$\beta_0 = \bar{y} - \beta_1 \bar{x} \qquad 4.9$$

$$\beta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \qquad 4.10$$

Dabei ist

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad 4.11$$

das arithmetische Mittel der x-Werte und $\bar{y}$ das arithmetische Mittel der y-Werte.

Wie man in der Formel 4.5 erkennen kann, werden Beobachtungen, die eine große Abweichung aufweisen, besonders stark gewichtet. Das bringt den Nachteil mit sich, dass die Methode der kleinsten Fehlerquadrate auf Ausreißer besonders stark reagiert. Anders ist das Verhalten bei der Summe der absoluten Abweichungen. Wie der Name schon vermuten lässt, werden statt der Fehlerquadrate die Absolutbeträge zur Schätzung der Parameter herangezogen. Diese werden im Vergleich zur MKQ deutlich geringer gewichtet, weshalb diese Methode deutlich robuster gegen Abweichungen ist.

4.5.2 „Stripe Analysis“

Bei der „Stripe Analysis“ handelt es sich um eine Clustering-Methode, die im vorliegenden Algorithmus eingesetzt wird, um den Datensatz aus weißen Pixeln vor der „Linearen Regression“ zu reduzieren. **Abbildung 4-4** zeigt das grundsätzliche Vorgehen. Wie bereits in Abschnitt 4.2. besprochen, werden Pflanzen bzw. Unkraut im Binärbild als weiße Pixel dargestellt. Ein weißes Pixel entspricht in der **Abbildung 4-4** einer 1. Für das vorliegende Beispiel wird ein Binärbild mit 9x9 Pixel mithilfe einer sogenannten „Sliding Box“ von 3x3 Pixel untersucht. Im ersten Schritt wird das Binärbild in horizontale Streifen aufgeteilt. Anschließend wird die „Sliding Box“ mit einem Inkrement von einem Pixel von links nach rechts verschoben. Bei jeder Position werden die weißen Pixel summiert und bei jener Box, bei der die Summe am größten ist, wird der Mittelpunkt markiert. Sollten mehrere Boxen in einer Zeile die gleiche Summe haben, wird der Median der Positionen bestimmt und der Mittelpunkt der jeweiligen Box markiert. Falls keine oder nur sehr wenige Einsen gefunden werden, wird der Mittelpunkt nicht markiert und die Box somit vernachlässigt. Dieser Vorgang wird für jeden horizontalen Streifen durchgeführt. Am Ende befindet sich die Box am unteren rechten Rand; unser Beispiel liefert 3 Mittelpunkte. Mithilfe der „Linearen Regression“ wird schlussendlich aus diesem reduzierten Datensatz eine Gerade gebildet, welche die Pflanzenreihe repräsentiert.

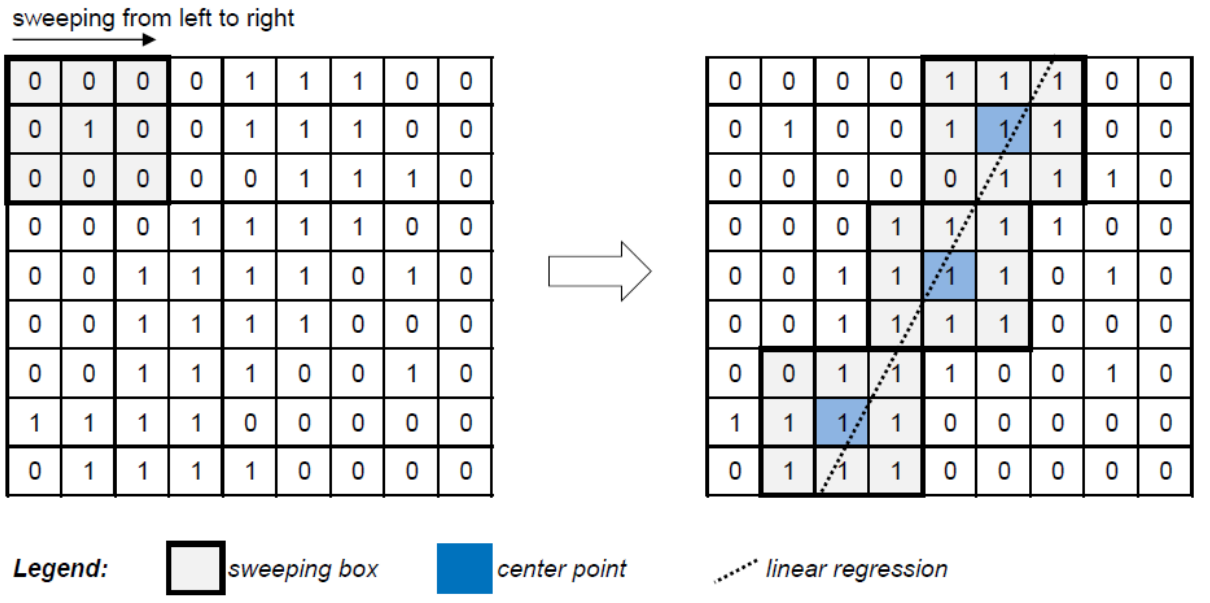


Abbildung 4-4: „Stripe Analysis“ für ein 9x9 Binärbild

Quelle: [9, Fig. 16]

4.6 Beurteilung der Simulationsergebnisse

Für die Beurteilung der Simulationsergebnisse wird die Software „Matlab“ verwendet. Mithilfe von „Matlab“ lassen sich die von der „Unreal Engine“ ausgegebenen Daten automatisiert einlesen, berechnen und grafisch darstellen. Im ersten Schritt wird daher ein Referenzverzeichnis auf neue, noch nicht berechnete Daten durchsucht. Anschließend werden die für die Berechnung relevanten „Ground Truth“ Daten und die Positionen des Implements während der Simulation eingelesen. Aus den „Ground Truth“ Daten lassen sich die Pflanzenreihen separieren und der Mittelwert der jeweiligen Pflanzenpaare ermitteln. Es wird jeweils der Pflanzenabstand in X- und Y-Richtung gemittelt. Diese Vorgehensweise setzt voraus, dass Pflanzen immer paarweise auftreten. In der Tat werden beinahe sämtliche Simulationen in der vorliegenden Arbeit mit derartigen Pflanzenpaaren simuliert. Lediglich in Abschnitt 7.1.3 werden einzelne Pflanzen ausgesetzt. Um die Methode zur Berechnung der Ideallinie dennoch anwenden zu können, werden die Positionen der Pflanzen in der „Unreal Engine“ unverändert festgelegt und erst im Nachhinein bestimmt, welche Pflanze in der Simulation hervorgebracht (engl. „to spawn“) werden sollen und welche nicht. In den Rohdaten werden sichtbare Pflanzen mit einer Eins und ausgeblendete mit einer Null gekennzeichnet. Unabhängig davon ob die Position der Pflanze mit einer Eins oder einer Null gekennzeichnet ist, wird die Ideallinie wie gehabt mittels „Matlab Skript“ berechnet. Ideal wäre es, wenn sich die Haken des Implements stets in der Mitte der Pflanzenreihen befinden. **Abbildung 4-5** stellt die Ermittlung der Ideallinie grafisch dar.

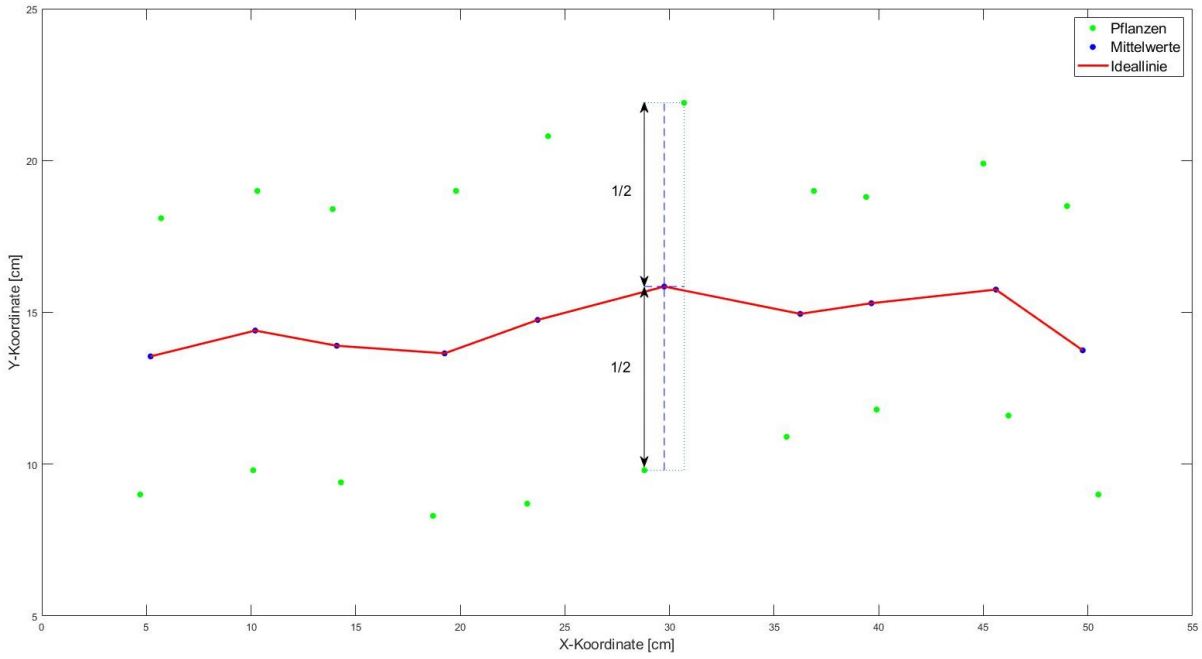


Abbildung 4-5: Berechnung der Ideallinie

Um die Simulationsergebnisse beurteilen zu können, wird die Abweichung der Ideallinie zur tatsächlich gefahrenen Linie berechnet. Die Informationen der tatsächlich gefahrenen Linie speichert die „Unreal Engine“ in der Datei „Track.txt“. Hierbei wird die aktuelle Position in äquidistanten Abständen in einen Textfile geschrieben und in „Matlab“ ausgewertet. Aufgrund des Offsets der einzelnen Pflanzen längs der Fahrtrichtung sind die Punkte, welche die Ideallinie beschreiben, nicht äquidistant. Es wird daher zwischen den einzelnen Punkten ein Polynom 1. Grades approximiert, das anschließend mit demselben Intervall wie die Positionsdaten abgetastet wird. Die Approximation erfolgt durch die „Matlabfunktion“ polyfit, die auf der bereits bekannten Methode der kleinsten Fehlerquadrate beruht (vgl. Abschnitt 4.5.1.3). Das Ergebnis der Abtastung dient dazu, die Abweichung zur tatsächlich gefahrenen Linie zu berechnen. Befindet sich das Implement genau auf der Ideallinie, so ist die Abweichung Null. Eine negative Abweichung bedeutet, dass sich das Implement zu weit rechts befindet, bei einer positiven Abweichung befindet es sich zu weit links. Der Mittelwert dieser Abweichungen wird mit der Formel 4.12 berechnet und bei den Messreihen in Kapitel 7 als „Mean deviation racing line“ bezeichnet. Das Ergebnis dieser Berechnung darf nicht als eine mittlere Abweichung betrachtet werden, da ein Vorzeichenwechsel das Ergebnis beeinflusst. Vielmehr sollte das Ergebnis Auskunft darüber geben, ob sich das Implement tendenziell zu weit links oder rechts der Ideallinie befindet. Um zusätzlich jedoch die tatsächliche mittlere Abweichung zu berechnen, werden alle Absolutbeträge der Abweichungen aufsummiert und durch deren Anzahl geteilt (vgl. Formel 4.13). Dieses Ergebnis wird als „Mean deviation racing line (ABS)“ bezeichnet. Mithilfe

der Formel 4.14 wird die maximale Abweichung berechnet („Maximum deviation racing line“). Da in dieser Formel der Absolutbetrag verwendet wird, kann ohne Rohdaten keine Aussage mehr darüber getroffen werden, in welche Richtung die Abweichung zur Ideallinie stattgefunden hat. Neben der mittleren und der maximalen Abweichung wird auch die Standardabweichung mit der Formel 4.15 berechnet. Die Standardabweichung gilt als Maß dafür, wie weit die Messabweichungen vom Mittelwert entfernt sind; sie wird in Kapitel 7 mit „Standard deviation racing line“ bezeichnet.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{4.12}$$

$$\mu_{ABS} = \frac{1}{N}\sum_{i=1}^{N} |A_i| \tag{4.13}$$

$$A_{max} = \max|A_i| \tag{4.14}$$

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |A_i - \mu|^2} \tag{4.15}$$

S … Standardabweichung
A_i … Abweichung zwischen Ideallinie und gefahrener Linie
A_{max} … Maximale Abweichung
N … Anzahl der Messabweichungen
μ … Mittlere Abweichung
μ_{ABS} … Mittlere Abweichung der Absolutbeträge

4.7 Präzision

„Präzision („precision“, nach DIN 55350): Qualitative Bezeichnung für das Ausmaß der gegenseitigen Annäherung voneinander unabhängiger Ermittlungsergebnisse bei mehrfacher Anwendung eines festgelegten Ermittlungsverfahrens unter vorgegebenen Bedingungen“ [31, p. 26].

„Wiederholpräzision ist die Präzision (Zeitabhängigkeit des Fehlers) unter wiederholbaren Bedingungen. Maß für die Übereinstimmung zwischen den Ergebnissen von unabhängigen Messungen derselben Messgröße. Dient der Charakterisierung einer Messmethode“ [31, p. 26].

„Richtigkeit („trueness, accuracy of the mean“, nach DIN 55350): Qualitative Bezeichnung für das Ausmaß der Annäherung des Erwartungswertes des Ermittlungsergebnisses an den Bezugswert,

wobei dieser je nach Festlegung oder Vereinbarung der wahre oder der richtige Wert sein kann“ [31, p. 26].

Um die Wiederholpräzision des vorliegenden „Prüfstandes“ zu untersuchen, wurde 10 Mal dieselbe Prüfung durchgeführt, um die Abweichungen untereinander festzustellen. **Abbildung 4-6** zeigt das Messergebnis dieser Messreihe. Wie ersichtlich, wird die Standardabweichung des Fehlers zwischen Ideallinie und gefahrener Linie ausgewertet, ferner der Mittelwert des Fehlers, die maximale Abweichung und die Standardabweichung des „Cross Track Errors“ (CTE).

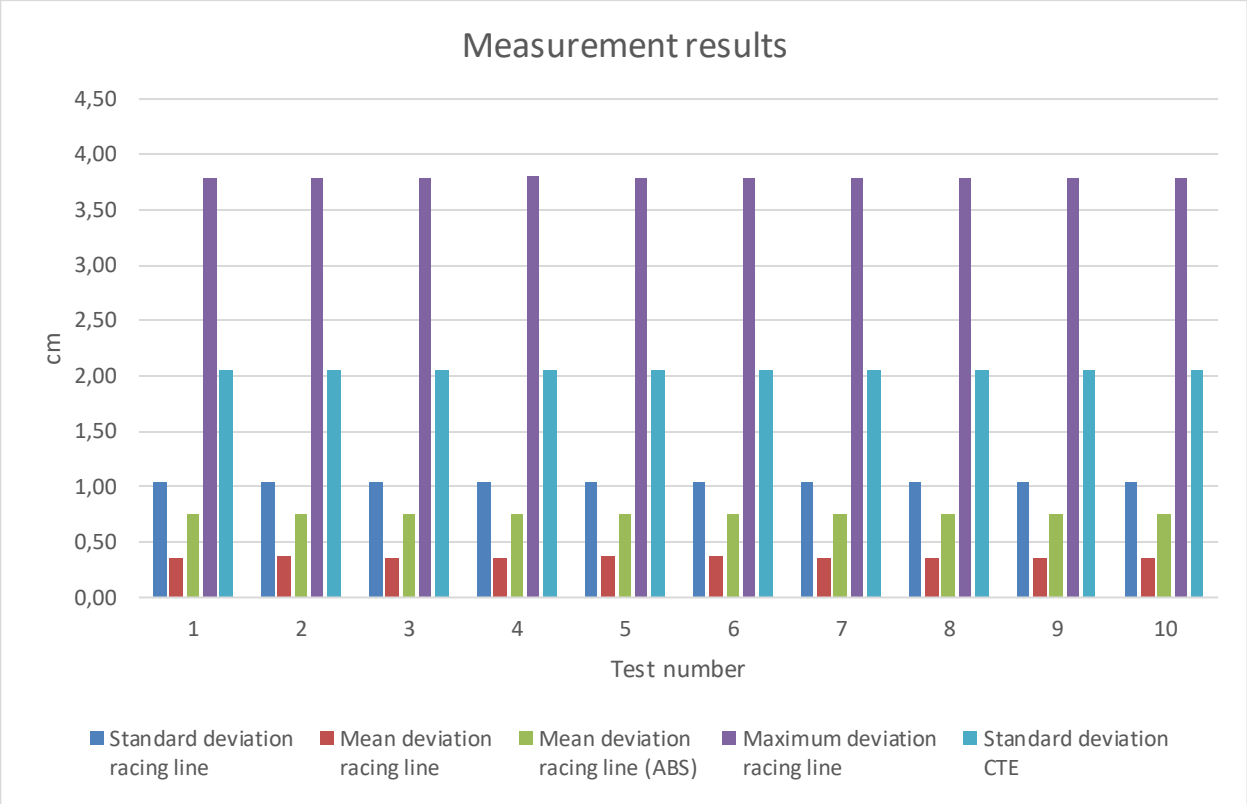


Abbildung 4-6: Messreihe zum Feststellen der Präzision

Bei Letzterem handelt es sich um die Abweichung zwischen Ideallinie und der aktuellen Position des Traktors [21, p. 30]. Dieser Fehler wird im bestehenden Algorithmus berechnet und anschließend an die „Unreal Engine“ gesendet. Der CTE ist der ausschlaggebende Parameter für die Lenkkorrektur. Ist er negativ, wird eine Lenkbewegung nach links durchgeführt, bei einem positiven CTE nach rechts. Der Betrag des CTE ist ein Maß für die Intensität der Lenkbewegung, diese kann mithilfe einer Konstante adjustiert werden. Der Nachteil des „Cross Track Errors“ liegt darin, dass er für die Berechnung der Ideallinie nicht die „Ground Truth“ Daten heranzieht, sondern diese aus dem Mittelwert der detektierten Pflanzenreihen berechnet. **Abbildung 4-7** stellt dieses Prinzip graphisch dar.

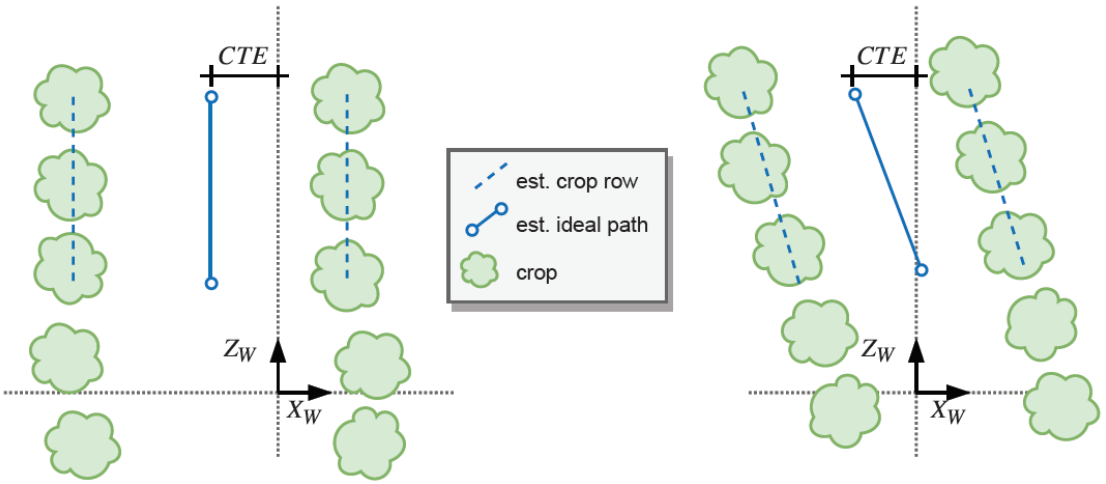


Abbildung 4-7: (links) CTE einer parallelen Ideallinie; (rechts) CTE einer schrägen Ideallinie

Quelle: [9, Fig. 17]

Tabelle 4-3 gibt Aufschluss über die Wiederholpräzision des „Prüfstandes". Es wird die Abweichung der einzelnen Messungen voneinander mithilfe der Standardabweichung bewertet. Wie man der **Tabelle 4-3** entnehmen kann, bewegen sich die Standardabweichungen zwischen 150nm und 98µm, was auf einen recht präzisen Prüfstand hinweist. Dass es aufgrund identer Simulationen weiterhin zu Messabweichungen kommt, liegt daran, dass die Synchronisierung zwischen der virtuellen Umgebung und dem Algorithmus nicht ideal ist. Totzeiten führen dazu, dass Lenkkorrekturen, wie sie in **Abbildung 4-1** zu sehen sind, nicht in Echtzeit übertragen werden.

Bezeichnung	Wert in cm
Standardabweichung der Standardabweichungen	0,000771
Standardabweichung der relativen Mittelwerte	0,000959
Standardabweichung der absoluten Mittelwerte	0,000525
Standardabweichung der Maximalwerte	0,009800
Standardabweichung der Standardabweichung CTE	0,000015

Tabelle 4-3: Abweichungen der einzelnen Messungen

5 DIE VIRTUELLE UMGEBUNG

5.1 Allgemein

Die virtuelle Umgebung in dieser Arbeit wurde in der „Unreal Engine 4“ entwickelt. Zum Zeitpunkt der Erstellung war diese Engine in der Version 4.24.0 verfügbar, die auch im Lauf des Projekts nicht mehr geändert wurde. Die „Unreal Engine 4“ (UE4) wurde von Epic Games entwickelt und zählt zu den modernsten und innovativsten Spiele Engines, die aktuell auf dem Markt verfügbar sind [28, p. 1]. Für Entwickler ist die Engine kostenlos, was sie für viele „open source“ Projekte besonders beliebt macht.

5.2 Anforderungen

Die Anforderungen, die an die virtuelle Umgebung gestellt wurden, sind vielseitig. Einerseits erwartet man sich eine virtuelle Umgebung, die der Realität möglichst nahekommt, und andererseits muss die Engine, mit der diese Umgebung erstellt wird, flexibel genug sein, um beispielsweise bestehende Projekte einzubinden. Im Marketplace der UE4 bieten sich unzählige Möglichkeiten, eine Umgebung so realistisch als möglich aussehen zu lassen. So können unterschiedlichste Texturen für den Ackerboden, diverse 3D-Modelle von Pflanzen, Tools für die Berechnung von Sonnenständen etc. teilweise kostenfrei heruntergeladen werden. Weiters bietet die UE4 eine „C++ application programming interface“ (API), die sich hervorragend eignet, um bestehende Projekte in die Engine einzubinden.

5.3 Erstellen der virtuellen Umgebung

5.3.1 „Blueprints“

Um die Anforderungen umzusetzen, bietet „Blueprints“ im Vergleich zu herkömmlichen C++ Codes eine einfache Alternative. Am besten lassen sich „Blueprints“ mit C++ Klassen vergleichen, nur dass anstelle von Codes mit sogenannten „nodes“ gearbeitet wird. Diese „nodes“ sind Bausteine, aus denen man unterschiedlichste Logiken zusammensetzen kann [28, p. 29]. Um das Prinzip zu verdeutlichen, stellt die **Abbildung 5-1** eine einfache Logik mittels „Blueprints“ dar.

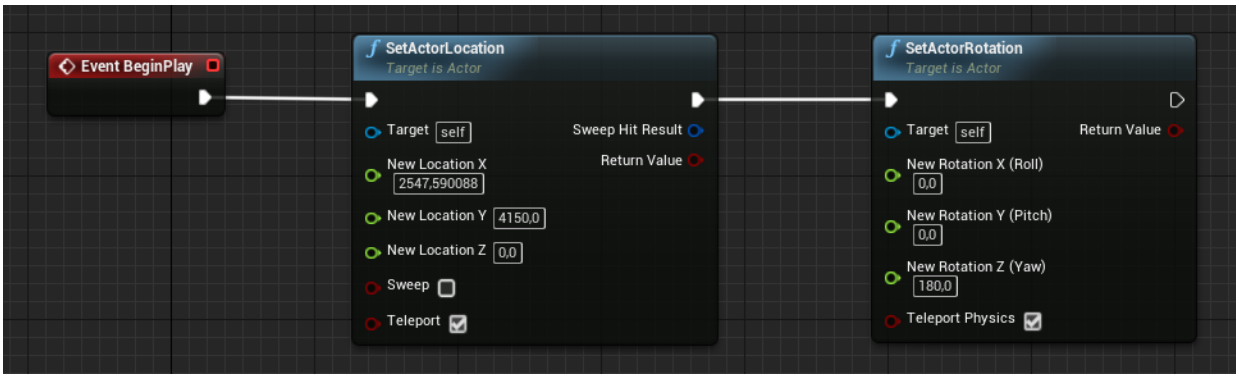


Abbildung 5-1: Funktionsprinzip „Blueprints“ mit 2 „nodes“

Ein „Actor“ wird hier, nachdem das Spiel gestartet wurde, an eine bestimmte Position am Spielfeld gesetzt und anschließend um 180 Grad um die Z-Achse gedreht. Unter einem „Actor“ versteht man in der „Unreal Engine“ ein Objekt, das in der Welt platziert oder hervorgebracht werden kann.

5.3.1.1 Automatisiertes Platzieren von Objekten

In dieser Arbeit werden „Blueprints“ und „Actors“ unter anderem dazu benutzt, um Pflanzen in der virtuellen Umgebung zu platzieren. Im eigens erstellten „Blueprint“ „Crop_Placement“ findet sich zum Beispiel eine Logik, die aus zwei verschachtelten „For-Schleifen“ besteht und Pflanzen in mehreren Reihen nach Wunsch platziert. Um auch hier der Realität so nahe wie möglich zu kommen, werden sowohl die Position als auch die Skalierung der Pflanzen mit einem einstellbaren Offset versehen. Die Skalierung der Pflanzen ermöglicht es dem Anwender, unterschiedlichste Wachstumsstadien zu simulieren.

Das Unkraut wird im „Blueprint“ „Weed_Placement“ platziert. Die Fläche des Ackers wird berechnet und das Unkraut mittels Zufallsgenerator verteilt. Die Option „Always Spawn, Ignore Collisions“ erlaubt es, dass Unkraut auch dann hervorgebracht werden kann, wenn die Position bereits durch eine Pflanze besetzt ist. **Abbildung 5-2** zeigt diesen Fall. Die Unkrautdichte kann vorab über den Paramater „Weed_Density“ in [Stück/m²] angegeben werden.

Abbildung 5-2: Pflanze und Unkraut an derselben Position

5.3.1.2 Steuerung der Kamera

Die Steuerung der Kamera und damit im weiteren Sinn auch die Steuerung des Traktors werden in der UE4 ebenfalls mit „Blueprints“ bewerkstelligt. Um die virtuelle Umgebung auch für zukünftige Projekte vielseitig einsetzen zu können, wurden zwei unterschiedliche „Blueprints“ entwickelt. Das „Blueprint“ „Camera_Implement“ spiegelt das aktuelle Setup wider. Wie eingangs erwähnt, wird der Traktor von einer Person gesteuert und die Algorithmen zur Reihenerkennung steuern das Implement. Die Steuerung des Traktors übernimmt in der „Unreal Engine“ ein „Spline“. „Splines“ spielen immer dann eine wichtige Rolle, wenn Kurven am Computer gezeichnet werden müssen [32]. Die Steuerung ist so ausgelegt, dass das Zugfahrzeug dem „Spline“ folgt und der Algorithmus zur Reihenerkennung lediglich die Bewegung des Implements, normal zur Fahrtrichtung, steuert. Beim zweiten „Blueprint“ „Tractor_Drive“ wird davon ausgegangen, dass die Kamera auf der Motorhaube bzw. vor dem Fahrzeug montiert ist. Dies ist notwendig, wenn Algorithmen für einen autonom fahrendes Fahrzeug getestet werden.

5.3.2 Plugins

Plugins sind eine Sammlung von Codes und Daten, welche die Entwickler nutzen, um neue Funktionen, Dateitypen und vieles mehr hinzuzufügen. Sie lassen sich problemlos und projektunabhängig aktivieren bzw. deaktivieren. Dieser Abschnitt stellt die in dieser Arbeit verwendeten Plugins in aller Kürze vor.

5.3.2.1 „Sun Position Calculator“

Mithilfe des „Sun Position Calculators“ ist es möglich, geographisch korrekte Sonnenstände in der virtuellen Umgebung zu simulieren. Damit erreicht man nicht nur eine realistisch wirkende Sonneneinstrahlung, sondern auch einen naturgetreuen Schattenwurf [33]. Für die Berechnung

der Position der Sonne werden Datum, Uhrzeit sowie geographische Koordinaten benötigt. In dieser Arbeit werden die Sonnenstände der Stadt Steyr in Oberösterreich berechnet. Steyr befindet sich 48° 3' 0.324" N 14° 25' 5.772" E (vgl. [34]), diese Koordinaten wurden in die „Unreal Engine“ übertragen.

5.3.2.2 „ROS Integration“

Das „ROS Integration Plugin“ ermöglicht es der „Unreal Engine“, mit einem laufenden „roscore“ zu kommunizieren. Unter einem „roscore“ versteht man eine Sammlung von „nodes“ und Programmen, welche Voraussetzung für ein ROS-basiertes System sind [35]. Das ROS Plugin unterstützt sowohl ROS „topics“ als auch ROS Services [36]. Die Kommunikation verläuft bidirektional. Daher ist es sowohl vom ROS als auch von der „Unreal Engine“ aus möglich, in „topics“ zu veröffentlichen (engl. „publish“) und sie zu abonnieren (engl. „subscribe“). Die Verbindung erfolgt über eine sogenannte „Rosbridge“. Die „Rosbridge“ stellt eine JSON („JavaScript Object Notation“) Programmierschnittstelle zur Verfügung, womit auch externe Programme auf das „Robot Operation System“ zugreifen können [37]. Um große Datenmengen wie „Videostreams“ übertragen zu können, wird die BSON („Binary JSON“) verwendet. Dabei handelt es sich um eine binär codierte Serialisierung von JSON [38].

5.3.2.3 „URoboVision“

Um einen „Videostream“ zwischen der „Unreal Engine“ und dem ROS aufzubauen, muss zusätzlich zum „ROS Integration Plugin“ das „URoboVision Plugin“ installiert werden. Dieses Plugin erlaubt es, mithilfe einer speziellen Kamera die RGB und Tiefenangaben (engl. „depth data“) von der „Unreal Engine“ in einer ROS-Umgebung zu veröffentlichen [36]. Die Kamera, die nach erfolgreicher Installation in der „Unreal Engine“ verfügbar ist, muss den „Blueprints“ „Camera_Implement“ und „Tractor_Drive“ als Kindklasse (engl. „Child class“) hinzugefügt werden.

5.3.3 Visual Effects

5.3.3.1 „Sky Light“

In der „Unreal Engine“ stimmen das Erscheinungsbild des Himmels sowie dessen Beleuchtung und Reflexion überein. Die Engine bietet mehrere Einstellungen, um das Himmelslicht dynamisch zu berechnen [39]. Da die Simulationszeiten in der vorliegenden Arbeit im Vergleich zu einem Tageszyklus relativ kurz sind, werden dynamische Vorgänge zur Berechnung des Himmelslichts deaktiviert. Durch die einmalige Berechnung des Himmelslichts am Anfang der Simulation wird nicht nur Rechenleistung eingespart, es ist auch im Sinne der Reproduzierbarkeit, wenn die Lichtverhältnisse während der Simulationsdauer konstant bleiben.

5.3.3.2 „Directional Lights“

Das „Directional Light“ (dt. gerichtetes Licht) simuliert Licht, das eine unendlich weit entfernte Quelle, beispielsweise die Sonne, ausstrahlt. Dies hat zur Folge, dass alle geworfenen Schatten parallel zueinander verlaufen [40]. Wie bereits beim „Sky Light“ in Abschnitt 5.3.3.1, sind auch beim „Directional Light“ die Einstellungen für eine dynamische Lichtberechnung deaktiviert.

5.3.3.3 „Sky Atmosphere“

In der „Unreal Engine“ dient die „Sky Atmosphere“ dazu, physikalisch korrekte Renderings vom Himmel und der Atmosphäre zu erzeugen [41]. Auch dieses System würde viele Möglichkeiten bieten, die virtuelle Umgebung noch realistischer wirken zu lassen. Da es aber das Ziel dieser Arbeit ist, wissenschaftliche reproduzierbare Ergebnisse hervorzubringen, werden auch hier sämtliche dynamischen Effekte deaktiviert.

5.3.3.4 „Exponential Height Fog“

Mithilfe des „Exponential Height Fog“ lässt sich in der „Unreal Engine“ realistischer Nebel simulieren. Die Dichte des Nebels ist von der Höhe des Geländes abhängig. An niedrigeren Stellen ist der Nebel dichter als an höheren. Durch Aktivierung des „Volumetric Fog“ wird das Licht im Nebel berechnet, was zu einer noch realistischeren Darstellung führt [42].

5.3.4 Einschränkungen

Wie bereits in Kapitel 3 erwähnt, wird eine virtuelle Umgebung (VR) niemals denselben Realitätsgrad erreichen wie die Natur. Auch die in dieser Arbeit vorgestellte VR kann diesen Grundsatz nicht aufheben. Dieser Abschnitt setzt sich daher mit den Einschränkungen auseinander, die zu akzeptieren sind, um das Verhältnis zwischen Aufwand und Nutzen in der Waage zu halten.

5.3.4.1 Die Pflanzen

Die virtuelle Umgebung, die im Zuge dieser Arbeit erstellt wurde, verwendet hochauflösende und sehr detaillierte 3D-Modelle aus dem, im Epic Store verfügbaren, Datensatz „Megascans – Lush Plants“. Zwar bietet dieser Datensatz dutzende Pflanzenmodelle an, doch ist die Auswahl an Kulturpflanzen, die in der Landwirtschaft angebaut werden, eher gering. Nichtsdestotrotz finden sich Pflanzen, die sich für die Evaluierung des Algorithmus eignen. Eine weitere Einschränkung dabei ist jedoch die fehlende Vielfalt dieser Pflanzen. Auf einer simulierten Strecke von 100 m, bei der die Pflanzen in zwei Reihen mit einem Pflanzenabstand von 15 cm positioniert sind, bekommt die Kamera mehr als 1300 Pflanzen vor die Linse (vgl. **Abbildung 5-3**). Diese Pflanzen werden zwar beliebig rotiert und skaliert, doch handelt es sich hierbei stets um dieselbe Pflanze mit derselben Farbe und derselben Geometrie. Einen deutlich realistischeren Ansatz verfolgen hier M. Cicco et al. [15]. Sie nutzen eine mehrschichtige radiale Verteilung der Blätter, um eine höhere Vielfalt und eine realistischere Darstellung der unterschiedlichen Wachstumsstadien der Pflanzen zu generieren.

5.3.4.2 Der Boden

Auch beim Boden müssen diverse Abstriche gegenüber der Realität in Kauf genommen werden. Beim Boden, der zum Großteil in dieser Arbeit verwendet wurde, handelt es sich um ein Standardmaterial der „Unreal Engine“, das „T_Metal_Rust_D“.

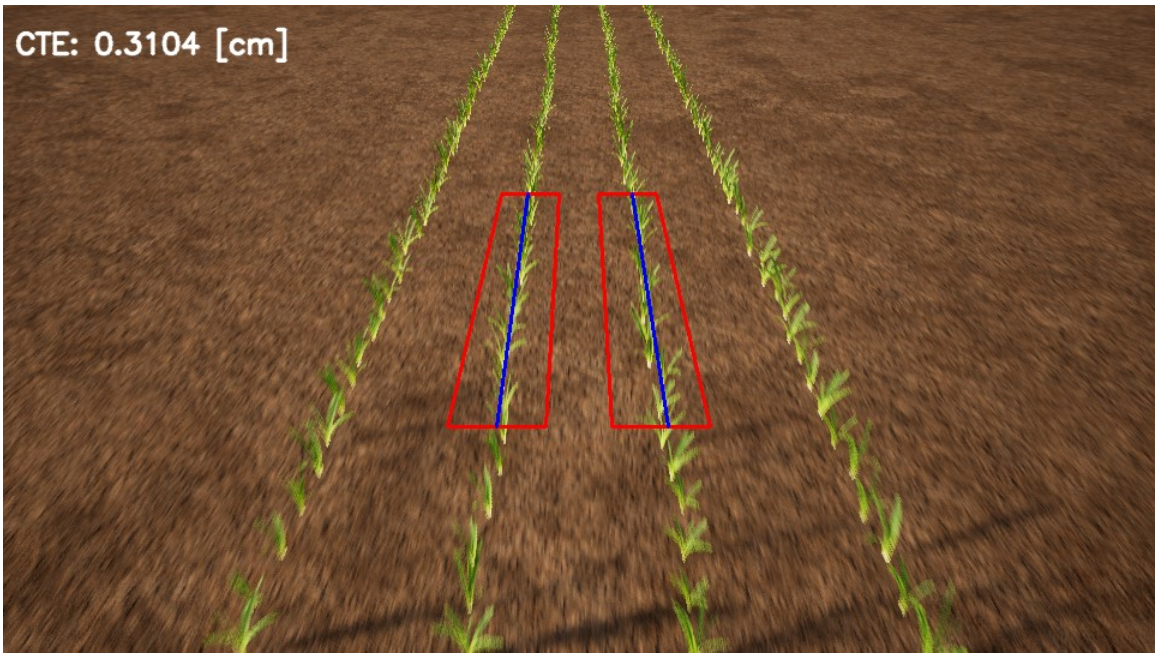


Abbildung 5-3: Virtuelle Umgebung

Wie in **Abbildung 5-3** zu sehen ist, weist dieser Boden keinen besonders hohen Detailgrad auf. Einen deutlich realistischeren Ansatz liefern wiederum M. Cicco et al. [15]. Das Forscherteam vermischt zwei unterschiedliche Texturen miteinander. Das Besondere an seinem Ansatz ist, dass er den „Lerp node“ (Lineare Interpolation) der „Unreal Engine“ nutzt. Dieser Knoten enthält zwei Eingänge für die Texturen und einen für die Intensität der Überblendung. An den Eingang für die Intensität hängen sie „Perlin-Noise“, was zu einer realistisch wirkenden Vermischung der beiden Texturen führt.

5.3.4.3 Fazit

Alle diese Einschränkungen mögen auf den ersten Blick negativ erscheinen, doch sollte man das Ziel dieser Arbeit nicht aus den Augen verlieren. Dieses Ziel ist es nämlich, einen „Software in the Loop“ Prüfstand zu entwickeln, mit dessen Hilfe ein bestehender Algorithmus zur Pflanzenreihen-Erkennung getestet und evaluiert werden kann. Dabei steht die wissenschaftliche Evaluierung eindeutig im Vordergrund. Eine virtuelle Umgebung, die alle Facetten der Natur wiedergibt, mag bei vielen Anwendungen von Vorteil sein, doch führen oft die einfachsten Simulationen zu wesentlichen Erkenntnissen. Daher soll eine gezielte Veränderung einiger weniger Parameter und die darauffolgende wissenschaftliche Evaluierung der Ergebnisse jenes Konzept sein, das diese Arbeit umsetzt.

6 DAS KAMERAMODELL

Tschentscher et al. [18] sehen die Herausforderung bei der Erstellung virtueller Umgebungen darin, diese so realistisch als möglich wirken zu lassen. Dies setzt einerseits realistische 3D-Modelle von Pflanzen und Boden voraus und andererseits ein detailliertes Modell jener Kamera, welche die Szene aufnimmt. Das Modell der virtuellen Kamera soll auf der Grundlage der Intel Realsense D435 erstellt werden.

6.1 Kameramodell der „Unreal Engine“

Grundsätzlich bietet die „Unreal Engine“ etliche Möglichkeiten, eine reale Kamera zu simulieren. Neben den Grundeinstellungen wie der Blende, der Belichtungszeit und dem ISO Wert lassen sich noch etliche Details wie der Weißabgleich, die Farbkorrektur und sogar die Zahl der Blendenlamellen einstellen. Eine Übersicht über die Einstellung bietet die „Unreal Engine“ Dokumentation [43]. Wie bereits in Abschnitt 5.3.2.3 erwähnt, verwendet diese Arbeit das „URoboVision Plugin“. Dieses erlaubt es leider nicht, alle Einstellungen direkt zu übernehmen. Statt der Abmessungen des Bildsensors und der Brennweite lässt sich beispielsweise beim „URoboVision Plugin“ nur mehr das Sichtfeld angeben. Mithilfe der Formel 6.1 und des Datenblatts des Bildsensors [44] ist dies allerdings mühelos zu berechnen.

$$\mathrm{HoV} = 2 * \arctan\left(\frac{\mathrm{B}}{2} * \frac{1}{\mathrm{F}}\right) \tag{6.1}$$

$$2 * \arctan\left(\frac{2{,}73\ \mathrm{mm}}{2} * \frac{1}{1{,}93\ \mathrm{mm}}\right) = 70{,}54° \tag{6.2}$$

HoV … „horizontal field of view“, Sichtfeld in Grad
B … Sensorbreite in mm
F … Brennweite in mm

Das klassische Lochkameramodell, wie es unter anderem in [45, Ch. 3] vorgestellt wird, beinhaltet intrinsische und extrinsische Parameter. Diese wiederum bieten in Summe zehn Freiheitsgrade. Bei den intrinsischen Parametern sind es zwei für die Verschiebung des Bildkoordinatensystems und jeweils einer für die horizontale und die vertikale Skalierung. Bei der extrinsischen Transformation fallen drei Parameter für die Rotation und drei für die Translation an. Mithilfe dieser Parameter lässt sich, über ein lineares Gleichungssystem, die dreidimensionale Welt auf der zweidimensionalen Bildebene der Kamera abbilden.

$$sm' = A[R|t]M' \tag{6.3}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13}\ t_1 \\ r_{21} & r_{22} & r_{23}\ t_2 \\ r_{31} & r_{32} & r_{33}\ t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6.4}$$

mit

A … Matrix der intrinsischen Koordinaten
R … Drehmatrix
t … Translationsvektor
X,Y,Z … Weltkoordinaten
M … Vektor der Weltkoordinaten
u,v … Bildkoordinaten in Pixel
f_x,f_y … Brennweite in Pixel
c_x,c_y … Hauptpunkt in Pixel
s … Skalierungsfaktor
m … Vektor der Bildkoordinaten

Da das Pixelseitenverhältnis des Bildsensors 1 beträgt und die Pixel somit quadratisch sind, gilt

$$f_x = f_y.$$

In der „Unreal Engine“ lassen sich nur die Parameter für die Translation explizit angeben. Die Rotation um die Y-Achse wird in Grad angegeben und intern in die Form einer Drehmatrix gebracht.

$$R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \tag{6.5}$$

Wie erwähnt, lässt sich im „URoboVision Plugin“ die Brennweite nicht frei wählen, sondern lediglich das Sichtfeld anpassen. Für die vorliegende Arbeit stellt das kein allzu großes Problem dar; allerdings müssen beim Kameramodell in der „Unreal Engine“ weitere Abstriche gemacht werden.

- Der Hauptpunkt der Kamera befindet sich stets in der Mitte und lässt sich nicht verändern.
- Radiale und tangentiale Verzerrungskomponenten können nicht angegeben werden, um beispielweise eine Linsenverzeichnung zu simulieren.
- Die Blende hat lediglich Auswirkungen auf die Tiefenschärfe, nicht aber auf die Belichtung.
- Der ISO Wert wirkt sich hingegen auf die Belichtung, nicht aber auf das Bildrauschen aus.
- Effekte wie die Bewegungsunschärfe oder das Bildrauschen lassen sich zwar simulieren, allerdings ist die Dokumentation dazu unvollkommen.

6.2 Kameramodell ROS

Das Kameramodell im ROS wurde bereits in einer früheren Arbeit [9, Ch. 3] ausführlich beschrieben, weshalb hier nur auf die Implementierung eingegangen wird. Damit der Reihendetektor ordnungsgemäß funktioniert, muss neben dem Translationsvektor, der Rotationsmatrix und der Bildauflösung auch die Brennweite in Pixel angegeben werden. Diese berechnet sich durch Einsetzen der Daten aus **Tabelle 6-1** und [44] wie folgt:

$$f_x = f_y = F * \frac{b}{B} \tag{6.6}$$

$$f_x = f_y = 1{,}88\ \text{mm} * \frac{960\ \text{px}}{2{,}73\ \text{mm}} \approx 679\ \text{px}. \tag{6.7}$$

F … Brennweite in mm
f_x,f_y … Brennweite in Pixel
B … Sensorbreite in mm
b … Sensorbreite in Pixel

Dass nicht mit der vollen Auflösung von 1920 X 1080 Pixel simuliert wird liegt daran, dass bereits in der Arbeit von R. Deutsch [9, Ch. 9.1] gezeigt werden konnte, dass eine Simulation mit maximaler Auflösung zu zu langen Rechenzeiten führt.

Abschließend muss noch das Koordinatensystem der „Unreal Engine“ mit jenem des „Robot Operating Systems“ abgeglichen werden. **Abbildung 6-1** zeigt das in dieser Arbeit implementierte Koordinatensystem. Die Indizes w stehen dabei für das Welt-Koordinatensystem und c für das Kamera-Koordinatensystem. Der Zusammenhang dieser beiden Koordinatensysteme wird durch die extrinsischen Parameter beschrieben.

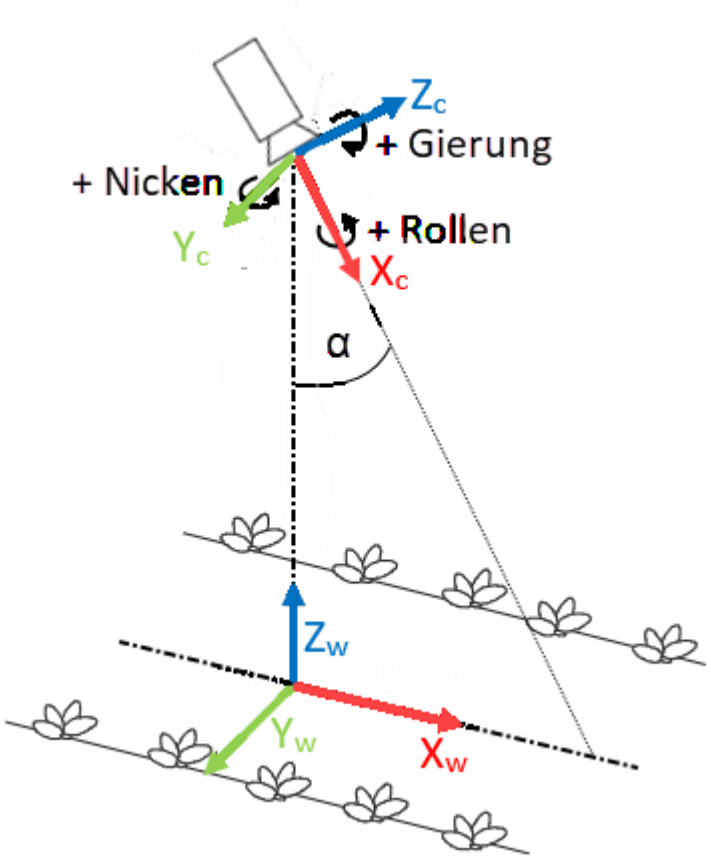


Abbildung 6-1: Position der Kamera auf dem Feld (modifizierte Abbildung aus [46], Farbschema „Unreal Engine“)

6.3 Kameraeffekte

6.3.1 Bewegungsunschärfe

H. Tümmel [47, p. 24] bezeichnet die Bewegungsunschärfe Δ_s als den Weg, der bei der Geschwindigkeit v des Objekts in der Belichtungszeit t bei gegebener Brennweite f und dem Abstand u zum Objekt entsteht.

$$\Delta_s = \frac{v * t * f}{u} \qquad 6.8$$

Die Bewegungsunschärfe nimmt daher mit zunehmender Belichtungszeit, größerer Brennweite und zunehmender relativer Bewegung zwischen Objekt und Kamera zu. Obwohl die Bewegungsunschärfe in Abschnitt 7.2.1 einige interessante Ergebnisse hervorbrachte, wird sie in der vorliegenden Arbeit als parasitärer Effekt angesehen. Im Gegensatz dazu wird sie unter anderem in der Sportfotografie dazu eingesetzt, um eine gewisse Dynamik im Bild zu erzeugen.

6.3.2 Rauschen

Rauschen ist ein Effekt, der in der Bildverarbeitung oft unerwünscht ist. Er kommt sowohl bei analogen als auch bei digitalen Geräten vor. Abhängig von der Anwendung unterscheidet man unterschiedliche Typen von Rauschen [48]. Das wohl bekannteste Rauschen ist das sogenannte weiße Rauschen – jenes Rauschen, das am häufigsten in Bildern vorkommt [49]. A. Erhardt [50, pp. 116–117] teilt das Rauschen unter anderem nach dessen Ursachen ein. Dazu zählen das Photonenrauschen, das Thermische Rauschen, das Ausleserauschen, das Verstärkerrauschen, das Quantisierungsrauschen sowie das Rauschen durch Inhomogenitäten des Kamerachips.

6.4 Reale Kamera Intel D435

Bei der Intel Realsense D435 handelt es sich um eine Stereokamera mit Infrarotprojektor und Farbsensor [51]. Sie wurde bereits in der Diplomarbeit „Vision-Based Crop Row Detection and Obstacle Recognition for Precision Farming“ [9, Ch. 7.2] ausgewählt. Die Kombination aus zwei monochromen Sensoren für die Stereobild-Erfassung und einem Farbsensor macht es möglich, diese Kamera sowohl für die Hinderniserkennung als auch für die Pflanzenreihen-Detektion einzusetzen. Der Infrarotsensor kommt in der Arbeit von R. Deutsch [9] nicht zum Einsatz, da es bei aktiven Kamerasystemen, die im Freien eingesetzt werden, zu Problemen kommen kann. Der Grund hierfür liegt darin, dass die Lichtquelle nicht mehr in der Lage ist, ausreichend Kontrast gegenüber dem Umgebungslicht zu erzeugen, was Messungen unmöglich macht [52]. Der Verschlusstyp „Rolling Shutter“ in **Tabelle 6-1** weist darauf hin, dass es sich beim Bildsensor „OmniVision OV2740“ um einen Sensor mit CMOS („complementary metal-oxide-semiconductor“) Sensor-Prinzip handeln muss. Im Gegensatz zur CCD („charge-coupled device“) Technologie ist hier die Ansteuerung einzelner Pixel möglich.

Die in der vorliegenden Arbeit verwendete Farbkamera hat einen „Rolling Shutter“. Dieser belichtet jede Zeile des CMOS-Sensors einzeln und liest diese anschließend sequentiell aus, was zu Bewegungsartefakten führen kann [53]. Im Gegensatz dazu haben jene Bildsensoren, welche für das Stereobild der Intel D435 verantwortlich sind, einen „Global Shutter“. Dieser ist für Stereosysteme notwendig, da diese die binokulare Perspektive nutzen um Tiefe zu rekonstruieren.

Parameter	Camera Sensor Properties
Image Sensor	OmniVision OV2740
Color Image Signal Processor	Discrete
Active Pixels	1920 X 1080
Sensor Aspect Ratio	16:09
Format	10-bit RAW RGB
F Number	f/2.0
Focal Length	1.93mm
Filter Type	IR Cut Filter
Focus	Fixed
Shutter Type	Rolling Shutter
Signal Interface	MIPI CSI-2, 1 Lane
Horizontal Field of View	69.4°
Vertical Field of View	42.5°
Diagonal Field of View	77.0°
Distortion	<=1.5%

Tabelle 6-1: Datenblatt Intel Realsense D435

Quelle: Intel® RealSense™ D400 Series Product Family Datasheet [51, p. 36]

Die Kamera wird werkseitig vor der Auslieferung kalibriert. Die Kalibrierdaten, zum Beispiel die Brennweite in Pixel, werden im „Robot Operating System“ in „topics“ (vgl. Abschnitt 4.3) zur Verfügung gestellt.

7 SIMULATIONEN

Wenn nicht anders angegeben, wurden die Messreihen mit folgenden Einstellungen durchgeführt. Die Kameraposition ist 1,5 m über dem Boden, der Nickwinkel α beträgt -30°. Die intrinsischen Kameraparameter entsprechen der Intel Realsense D435 [54]. Da sich die Simulation auf keine spezielle Fruchtsorte beschränkt, ist der Abstand der Pflanzen untereinander sowohl in X- als auch in Y-Richtung 50 cm ± Offset. Reale Messungen wurden auf einem Maisfeld durchgeführt, der Reihenabstand (Y-Richtung) betrug hier 50 cm ± Offset und der Pflanzenabstand (X-Richtung) 15 cm ± Offset. Messreihen, bei denen der Abstand eine signifikante Rolle spielt, wurden daher sowohl mit 50 als auch mit 15 cm Pflanzenabstand durchgeführt. Die Geschwindigkeit des Traktors betrug bei allen Messreihen 2m/s. Die Bildfrequenz des „Videostreams“ war mit 15 FPS konstant.

Um die unterschiedlichen Messreihen miteinander vergleichen zu können, ist das grundlegende Setup für alle Messreihen gleich. Sofern nicht anders angegeben, wird ein Acker, dessen geographische Koordinaten sich in Steyr befinden, am 1. Juni um 7 Uhr morgens simuliert. Die Simulation wird so eingerichtet, als würde die Zugmaschine inkl. Implement gegen Süden fahren. Für die Pflanzen wird mittels Skalierung ein Wachstumsstadium zwischen 20% und 30% angenommen und die Rotation liegt zwischen 0° und 360°. Da die „Unreal Engine“ aber unzählige Einstellungen bietet und einige davon signifikante Einflüsse auf das Messergebnis haben, sollten immer nur Messergebnisse innerhalb eines Abschnitts miteinander verglichen werden. Es wird versucht, in jedem Abschnitt nur die notwendigsten Parameter zu variieren, um die Auswirkungen auf das Messergebnis zu verdeutlichen.

Auf eine genaue Bewertung ab wann ein Messergebnis gut/schlecht, positiv oder negativ ist, soll in dieser Arbeit verzichtet werden. Grund dafür ist der universell einsetzbare Algorithmus, welcher nicht nur zum Jäten sondern auch für andere mechanische Verfahren wie das Pflügen, Fräsen, Grubbern, Eggen und Hacken eingesetzt werden kann [55, p. 1227]. Alle diese Verfahren erfordern unterschiedliche Toleranzen, weshalb eine Festlegung auf eine fest vorgegebene Grenze keine Vorteile mit sich bringt.

7.1 „Lineare Regression“ vs. „Stripe Analysis“

In diesem Abschnitt geht es darum, herauszufinden, welches Approximationsverfahren besser für die Reihenerkennung geeignet ist. Es werden dabei sowohl mit der „Linearen Regression“ (vgl. Abschnitt 4.5.1) als auch mit der „Stripe Analysis“ (vgl. Abschnitt 4.5.2) vier Messreihen durchgeführt.

7.1.1 Messreihe ohne Pflanzenoffset

7.1.1.1 Beschreibung

Bei der ersten Messreihe werden die Pflanzen ohne Offset im Abstand von 15 cm positioniert. Aufgrund des fehlenden Offsets entspricht die Ideallinie einer Geraden (siehe **Abbildung 7-2**). Dieser Messaufbau entspricht nicht der Realität, er dient lediglich dazu, die beiden Approximationsverfahren ohne weitere Störeinflüsse zu evaluieren.

7.1.1.2 Ergebnisse

Wie in **Abbildung 7-1** zu sehen ist, liefert die „Lineare Regression“ bei dieser Messreihe deutlich bessere Ergebnisse. Zwar sind die Mittleren Abweichungen zur Ideallinie mit deutlich unter 1 cm für beide Approximationsverfahren sehr gut, doch ist dieser Wert bei der „Stripe Analysis“ doch um 50% höher als bei der „Linearen Regression“.

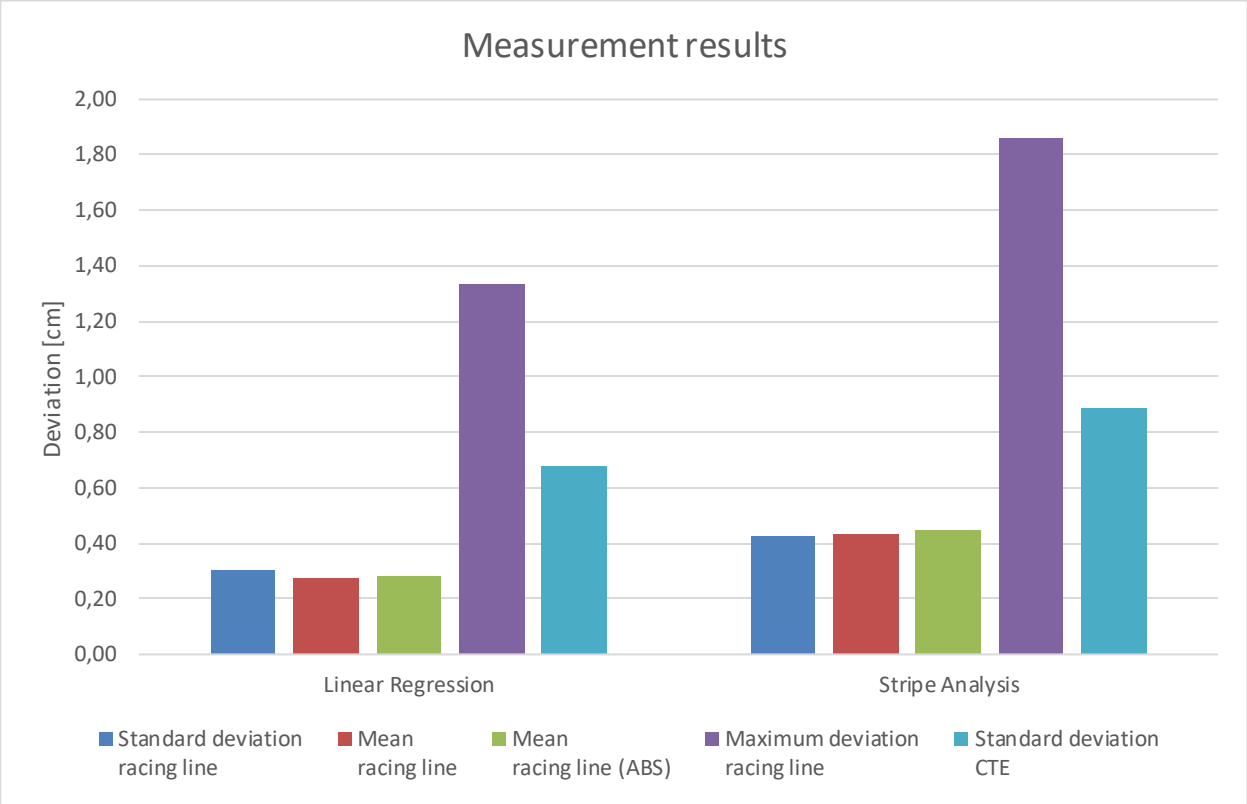


Abbildung 7-1: Messergebnisse „Lineare Regression“ vs. „Stripe Analysis“

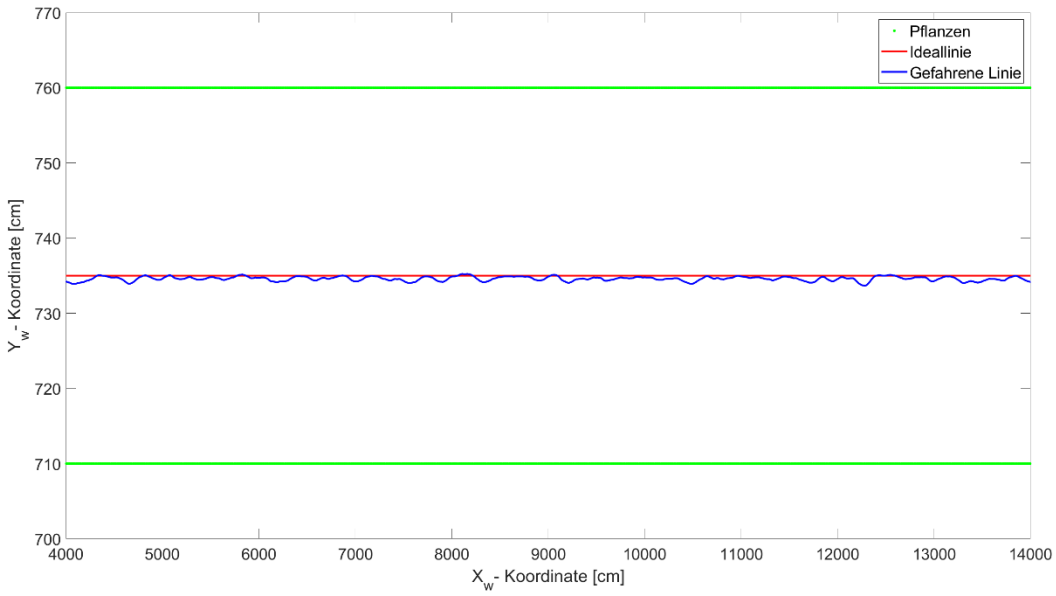


Abbildung 7-2: Messergebnis mit „Linearer Regression“

7.1.2 Messreihe mit Pflanzenoffset

7.1.2.1 Beschreibung

Hier handelt es sich um eine Messreihe, bei der die Positionen der Pflanzen mit einem Offset von ± 2 cm beaufschlagt werden. Dieses Setup entspricht im Vergleich zu Abschnitt 7.1.1 Messreihe ohne Pflanzenoffset schon eher der Realität. Um jedoch weiterhin die beiden Approximationsverfahren vergleichen zu können, wird auf weitere Veränderungen im Setup verzichtet, um das Messergebnis nicht unnötig zu verfälschen.

7.1.2.2 Ergebnisse

Abbildung 7-3 vergleicht die Messergebnisse der „Linearen Regression“ mit denen der „Stripe Analysis“. Wie schon im vorangegangenen Abschnitt, liefert die „Lineare Regression“ auch bei einem Pflanzenoffset ein besseres Ergebnis.

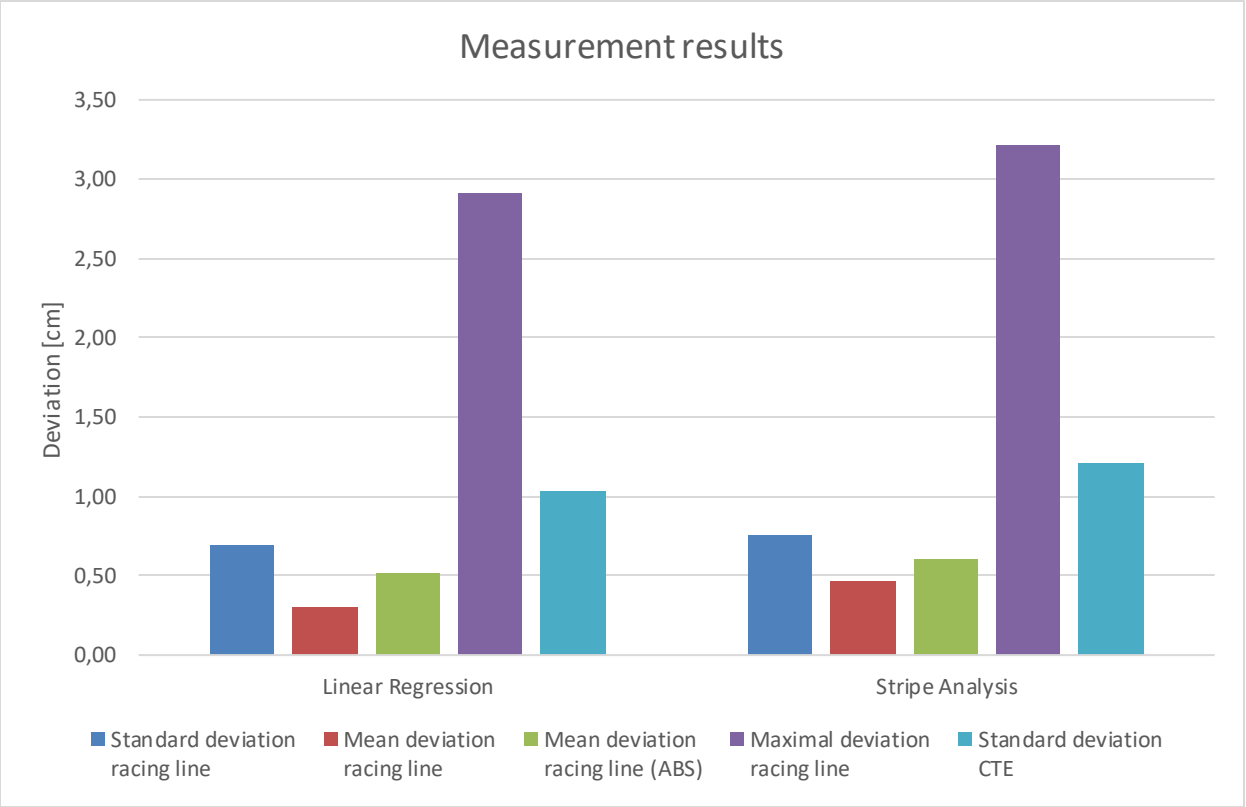


Abbildung 7-3: Messergebnisse „Linearer Regression“ vs. „Stripe Analysis“ mit Offset

7.1.3 Messreihe mit Aussetzern in der Pflanzenreihe

7.1.3.1 Beschreibung

Mithilfe dieser Messreihe soll evaluiert werden, welches der beiden Approximationsverfahren sich besser für unregelmäßige Pflanzenreihen eignet. Dazu werden Pflanzenaussetzer in der Reihe simuliert. Die „Unreal Engine“ bietet die Möglichkeit, wiederholbare Zufallszahlen innerhalb eines zuvor festgelegten Bereichs zu generieren. Diese Möglichkeit wird zur Ausblendung einzelner, scheinbar willkürlich ausgewählter Pflanzen genutzt. Es wird in der Engine ein „Seed“ (dt. Samen) gesetzt, der die Zufallsverteilung festlegt. Dies hat den Vorteil, dass Messungen mit identer Zufallsverteilung miteinander verglichen und beliebig oft reproduziert werden können. Da davon auszugehen ist, dass sich das Messergebnis mit zunehmendem Pflanzenabstand verändert, wird diese Messreihe sowohl mit 15 ± 2 cm als auch mit 50 ± 2 cm Pflanzenabstand durchgeführt.

7.1.3.2 Ergebnisse

Wie schon in den vorangegangenen Abschnitten 7.1.1 und 7.1.2, liefert die „Lineare Regression“ (LR) auch bei Aussetzern in der Pflanzenreihe deutlich bessere Ergebnisse als die „Stripe Analysis“ (SA). **Tabelle 7-1** fasst die Ergebnisse für die Messreihen mit 15 ± 2 cm Pflanzenabstand zusammen. Mit einer durchschnittlichen Abweichung von knapp über 1 cm und einer maximalen Abweichung von rund 5 cm bei der „Stripe Analysis“ kann man die Resultate jedoch immer noch als zufriedenstellend bezeichnen.

Measu-rement number	Name	Value	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE	
			LR	SA	LR	SA	LR	SA	LR	SA	LR	SA
1	Dropout	1/10	0,70	0,78	0,30	0,49	0,52	0,63	2,91	3,62	0,98	1,19
2	Dropout	1/9	0,70	0,79	0,30	0,51	0,52	0,64	2,85	3,72	0,98	1,19
3	Dropout	1/8	0,70	0,79	0,28	0,51	0,52	0,64	2,85	3,81	1,00	1,20
4	Dropout	1/7	0,70	0,79	0,29	0,54	0,52	0,66	2,85	3,56	0,99	1,21
5	Dropout	1/6	0,71	0,80	0,27	0,53	0,52	0,66	2,85	3,56	1,01	1,24
6	Dropout	1/5	0,72	0,82	0,27	0,56	0,53	0,69	2,88	3,69	1,03	1,30
7	Dropout	1/4	0,73	0,88	0,29	0,63	0,54	0,75	3,06	4,12	1,05	1,40
8	Dropout	1/3	0,79	0,95	0,23	0,72	0,56	0,83	3,19	4,50	1,15	1,51
9	Dropout	1/2	0,82	1,13	0,35	1,06	0,60	1,12	4,03	5,15	1,55	1,94

Tabelle 7-1: Messergebnis Aussetzer in der Pflanzenreihe 15 ± 2 cm Pflanzenabstand

Um die Robustheit der beiden Verfahren genauer zu untersuchen, wird der Pflanzenabstand von 15 ± 2 cm auf 50 ± 2 cm erhöht, was in etwa einer Kürbiskultur entspricht. **Abbildung 7-4** zeigt das Messergebnis für eine Simulation mit „Linearer Regression“, bei der 1 von 9 Pflanzen ausgeblendet wird. Der Algorithmus ist bereits nach etwa 47 m nicht mehr imstande, die Spur zu halten. Dasselbe Verhalten ist auch bei der „Stripe Analysis“ anzutreffen.

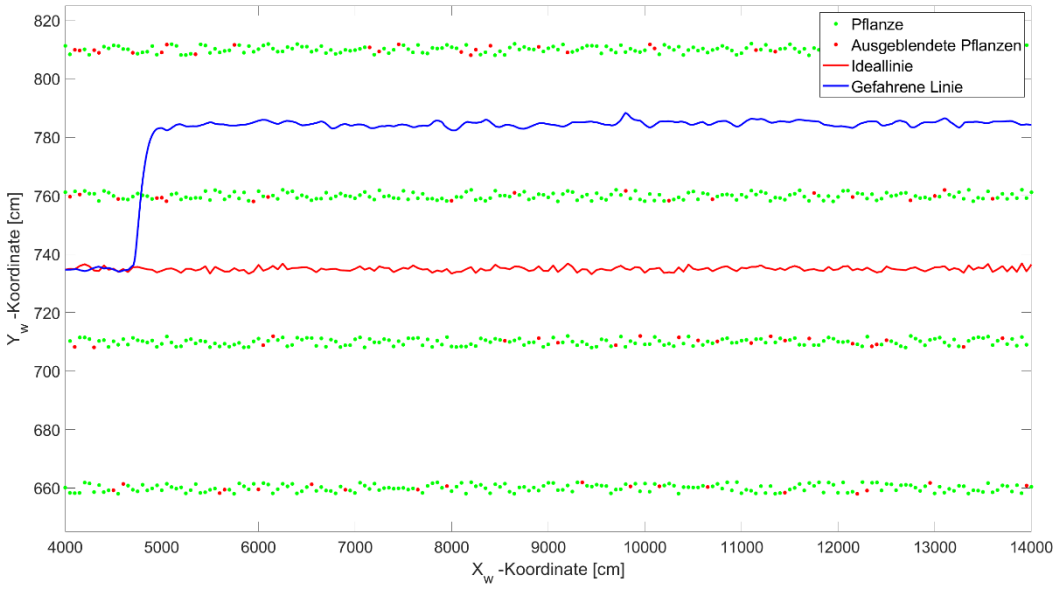


Abbildung 7-4: Simulation Aussetzer in der Pflanzenreihe 50 ± 2 cm Pflanzenabstand (LR)

Da auch diese Simulation keine Aussage darüber erlaubt, welches der beiden Verfahren sich besser für das Detektieren unvollständiger Pflanzenreihen eignet, wird in **Tabelle 7-2** noch das Ergebnis einer dritten Messreihe mit 50 ± 2 cm Pflanzenabstand vorgestellt. Bei dieser werden mehrere Versuche mit jeweils 10 unterschiedlichen Zufallsverteilungen durchgeführt. Die Verteilung der ausgeblendeten Pflanzen ist sowohl für die LR als auch für die SA ident. Es wurden in etwa 10% der Pflanzen ausgeblendet.

Measu-rement number	Name	Value	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE		Status	
			LR	SA	LR	SA	LR	SA	LR	SA	LR	SA	LR	SA
1	Seed	1	0,86	1,40	0,42	1,66	0,68	1,66	3,33	5,22	1,46	1,94	pass	pass
2	Seed	2	0,84	1,41	0,40	1,67	0,63	1,68	3,87	5,98	1,63	2,04	pass	pass
3	Seed	3		1,52		1,68		1,71		6,02		2,39	fail	pass
4	Seed	4	0,84	1,41	0,43	1,68	0,65	1,68	3,36	5,50	1,45	1,81	pass	pass
5	Seed	5	-	-	-	-	-	-	-	-	-	-	fail	fail
6	Seed	6	-	-	-	-	-	-	-	-	-	-	fail	fail
7	Seed	7	0,89	1,42	0,47	1,69	0,69	1,70	3,80	5,57	1,83	2,09	pass	pass
8	Seed	8	0,88	1,44	0,41	1,68	0,67	1,69	3,46	5,63	1,55	1,89	pass	pass
9	Seed	9	-	-	-	-	-	-	-	-	-	-	fail	fail
10	Seed	10	0,96	-	0,40	-	0,70	-	4,05	-	1,59	-	pass	fail

Tabelle 7-2: Messergebnis Aussetzer der Pflanzenreihe 50 ± 2 cm Pflanzenabstand und variierende Verteilung

Vergleicht man diese Messergebnisse aus **Tabelle 7-2**, stellt man fest, dass mithilfe des Approximationsverfahrens „Lineare Regression“ deutlich geringere Abweichungen zwischen Ideallinie und gefahrener Linie erreichbar sind. Ein Problem haben beide Verfahren jedoch gemeinsam: Lediglich 60% der Messungen konnten bis zum Ende durchgeführt werden. Die Messungen 5, 6 und 9 konnten mit beiden Verfahren nicht erfolgreich beendet werden, bei der 3. Messung scheiterte lediglich die „Lineare Regression“ und bei der 10. Messung nur die „Stripe Analysis“.

7.1.4 Messreihe mit unterschiedlichen Pflanzen

7.1.4.1 Beschreibung

Um festzustellen wie der Algorithmus auf verschiedenste Pflanzentypen reagiert, werden in dieser Messreihe Simulationen mit 3 unterschiedlichen Pflanzen durchgeführt (siehe **Abbildung 7-5**). Bei dieser Messreihe wird zusätzlich die „Blob reduction“, eine Erweiterung des Algorithmus, um ein besseres Verhalten bei starkem Unkrautbewuchs zu gewährleisten (vgl. Abschnitt 7.5.2), eingesetzt.

Abbildung 7-5: Pflanze 1 (Links), Pflanze 2 (Mitte), Pflanze 3 (Rechts)

7.1.4.2 Ergebnisse

Tabelle 7-3 zeigt die Ergebnisse der Simulation mit unterschiedlichen Pflanzen. Auffällig ist, dass der Algorithmus mit zugeschalteter „Blob reduction“ bei der Pflanze 1 nicht in der Lage ist, die Simulationen ordnungsgemäß abzuschließen. Es kommt zu mehreren Spurwechseln, welche die Abweichungen exorbitant ansteigen lassen. Grund dafür sind die dünnen Strukturen von Pflanze 1 (vgl. **Abbildung 7-8**).In **Tabelle 7-3** werden diese Ergebnisse nicht dargestellt, da sie in keinem Verhältnis zu den anderen stehen. Eine weitere Anomalie befindet sich in den Messergebnissen für Pflanze 2. Obwohl die Standardabweichung CTE bei der SA mehr als doppelt so groß ist als bei der LR, spiegelt sich dieses Ergebnis nicht in der Standardabweichung der Ideallinie wider. Das ist auf die Geometrie der Pflanze 2 zurückzuführen. Diese füllt beinahe die komplette „Region of Interest“ weshalb der Algorithmus versucht, mittels CTE Korrekturen mit hoher Frequenz vorzunehmen. Das Implement kann aufgrund seiner Trägheit dieser hohen Frequenz nicht folgen, was dazu führt, dass die Standardabweichung der Ideallinie nicht mit jener der CTE korreliert.

Plant	Blob reduction	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE	
		LR	SA	LR	SA	LR	SA	LR	SA	LR	SA
1	on	-	-	-	-	-	-	-	-	-	-
2	on	1,36	1,33	1,50	-0,14	1,53	0,95	5,02	4,23	1,60	3,28
3	on	1,46	1,60	0,38	0,37	1,01	1,15	5,51	5,60	3,41	3,98
1	off	0,84	1,10	0,05	0,91	0,57	1,03	2,64	3,58	1,79	2,04
2	off	1,38	1,38	1,57	-0,20	1,60	0,99	5,24	4,46	1,54	3,42
3	off	1,22	1,53	0,48	0,44	0,88	1,09	4,73	5,15	2,74	3,76

Tabelle 7-3: Messergebnis Variation der Pflanzen

7.1.5 Messreihe mit Unkraut

7.1.5.1 Beschreibung

Wie bereits in Abschnitt 4.5 beschrieben haben sowohl die „Lineare Regression“ als auch die „Stripe Analysis“ ihre Vor- und Nachteile. Beispielsweise soll die „Lineare Regression“ aufgrund der starken Gewichtung von größeren Abweichungen besonders empfindlich auf Ausreißer reagieren. Diese Messreihe vergleicht daher die LR und die SA, wenn derartige Ausreißer in Form von Unkraut simuliert werden. Da es sich hierbei um eine Simulation mit Unkraut handelt wird die „Blob reduction“ im Algorithmus für diese Messreihe aktiviert.

7.1.5.2 Ergebnisse

Um die Messergebnisse beider Approximationsverfahren vergleichen zu können, werden diese in **Tabelle 7-4** anschaulich dargestellt. Eine graphische Darstellung der Ergebnisse der „Linearen Regression“ findet sich in Abschnitt 7.5.2. Sowohl die **Tabelle 7-4** als auch die **Abbildung 7-48** aus Abschnitt 7.5.2 kann den Nachteil der Anfälligkeit der LR gegenüber Ausreißern nicht bestätigen. Die Messergebnisse sind beinahe über die gesamte Messreihe sowohl bei der LR als auch bei der SA konstant. Wie schon in den Abschnitten 7.1.1 bis 7.1.3, liefert die „Lineare Regression“ bessere Ergebnisse als die „Stripe Analysis“.

Measurement Number	Name	Value	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE	
			LR	SA	LR	SA	LR	SA	LR	SA	LR	SA
1	Weed density	0	0,72	0,77	0,32	0,48	0,54	0,63	2,70	3,25	1,01	1,23
2	Weed density	2	0,72	0,78	0,31	0,49	0,54	0,64	2,75	3,34	1,02	1,26
3	Weed density	4	0,72	0,78	0,32	0,47	0,54	0,63	2,73	3,31	1,02	1,24
4	Weed density	6	0,72	0,77	0,30	0,46	0,54	0,62	2,67	3,22	1,04	1,24
5	Weed density	8	0,72	0,76	0,29	0,44	0,54	0,61	2,61	3,23	1,06	1,23
6	Weed density	10	0,73	0,77	0,30	0,43	0,55	0,61	2,72	3,13	1,07	1,23
7	Weed density	12	0,73	0,77	0,29	0,44	0,54	0,62	2,67	3,00	1,08	1,26
8	Weed density	14	0,73	0,77	0,30	0,44	0,55	0,61	2,61	2,94	1,11	1,25
9	Weed density	16	0,72	0,76	0,28	0,41	0,54	0,60	2,67	3,06	1,10	1,24
10	Weed density	18	0,72	0,75	0,31	0,41	0,54	0,60	2,61	3,06	1,09	1,24
11	Weed density	20	0,72	0,75	0,31	0,40	0,54	0,59	2,60	3,10	1,12	1,23

Tabelle 7-4: Messergebnis Variation der Unkrautdichte

7.1.6 Analyse der Approximationsverfahren

In den Abschnitten 7.1.1 bis 7.1.5 wurden mithilfe der „Linearen Regression“ sowie die „Stripe Analysis“ umfangreichen Simulationen durchgeführt. Als Einführung begann Abschnitt 7.1.1 mit einer Messreihe, bei welcher die Pflanzen zwar zwischen 0° und 360° Grad rotiert wurden, auf einen zusätzlichen Pflanzenoffset wurde aber vorerst verzichtet. Dass die gefahrene Linie in **Abbildung 7-2** relativ kurvig ist, ist der Rotation der Pflanzen geschuldet. Messungen, welche ohne Pflanzenrotation durchgeführt werden, weisen eine deutlich geradere Linie auf (vgl. Abschnitt 7.4.2). Um den Realitätsgrad weiter zu steigern, wurden die Pflanzen in Abschnitt 7.1.2 mit einem Offset von ± 2 cm beaufschlagt. Bereits nach diesen beiden Messreihen konnte festgestellt werden, dass die LR im Vergleich zu SA bessere Ergebnisse liefert. Anschließend wurden in Abschnitt 7.1.3 mehrere Messungen durchgeführt bei welchen Aussetzer in der Pflanzenreihe simuliert wurden. Aufgrund der Tatsache, dass sowohl die „Lineare Regression“ als auch die „Stripe Analysis“ Simulationen mit einem Pflanzenabstand von 15 ± 2 cm und einer

Ausfallrate von 50% problemlos durchführen konnten, wurde der Pflanzenabstand auf 50 ± 2 cm erhöht. Mit dem erhöhten Pflanzenabstand wurden Simulationen durchgeführt, bei welchen eine von neun Pflanzen ausgesetzt wird. Durch eine zufällige Variation der ausgesetzten Pflanzen konnte festgestellt werden, dass beide Verfahren nicht mehr in der Lage sind die Simulation zu Ende zu führen wenn 3 Pflanzen hintereinander ausgeblendet sind (vgl. **Abbildung 7-4**). Bei einem Pflanzenabstand von rund 50 cm entsprechen 3 ausgesetzte Pflanzen einer Lücke von rund 150 cm. Die voreingestellte Länge der „Region of Interest" beträgt 2 m was bedeutet, dass bei einem Aussetzer von 3 Pflanzen lediglich eine sichtbar ist. Bekannterweise benötigt man für die Darstellung einer Geraden 2 Punkte. In Abschnitt 7.3.6 wird zwar festgestellt, dass eine durchschnittliche Pflanze aus 57 weißen Pixeln besteht, doch können diese nur Informationen zur Pflanze selbst und nicht zur Reihe liefern. Weder die LR noch die SA können diesen Unterschied feststellen, weshalb die approximierte Linie in so einem Fall von der Geometrie und der Ausrichtung einer Pflanze abhängt. Das führ unweigerlich zu einem Verhalten wie es in **Abbildung 7-4** zu sehen ist. Der Winkel zwischen Kamera und detektierter Reihe ist stets bekannt. Legt man eine maximal zulässige Änderung dieses Winkels fest, könnte das dem Problem entgegenwirken.

In Abschnitt 7.1.4 wurden Messreihen mit drei unterschiedlichen Pflanzen simuliert. Auf die Besonderheiten der Messergebnisse mit Pflanze 2 wurde bereits in Abschnitt 7.1.4.2 eingegangen, doch soll die **Abbildung 7-6** noch einmal die Unterschiede der „Linearen Regression" und der „Stripe Analyse" aufzeigen. Betrachtet man die Ergebnisse aus **Tabelle 7-3** welche ohne „Blob reduction" durchgeführt wurden, stellt man fest, dass die LR bei Pflanze 1 und 3 eindeutig bessere Ergebnisse liefert. Diese Ergebnisse lassen sich auch aus **Abbildung 7-6** interpretieren auch wenn der Unterschied aufgrund der langen Strecke nicht ganz so eindeutig hervorgeht.

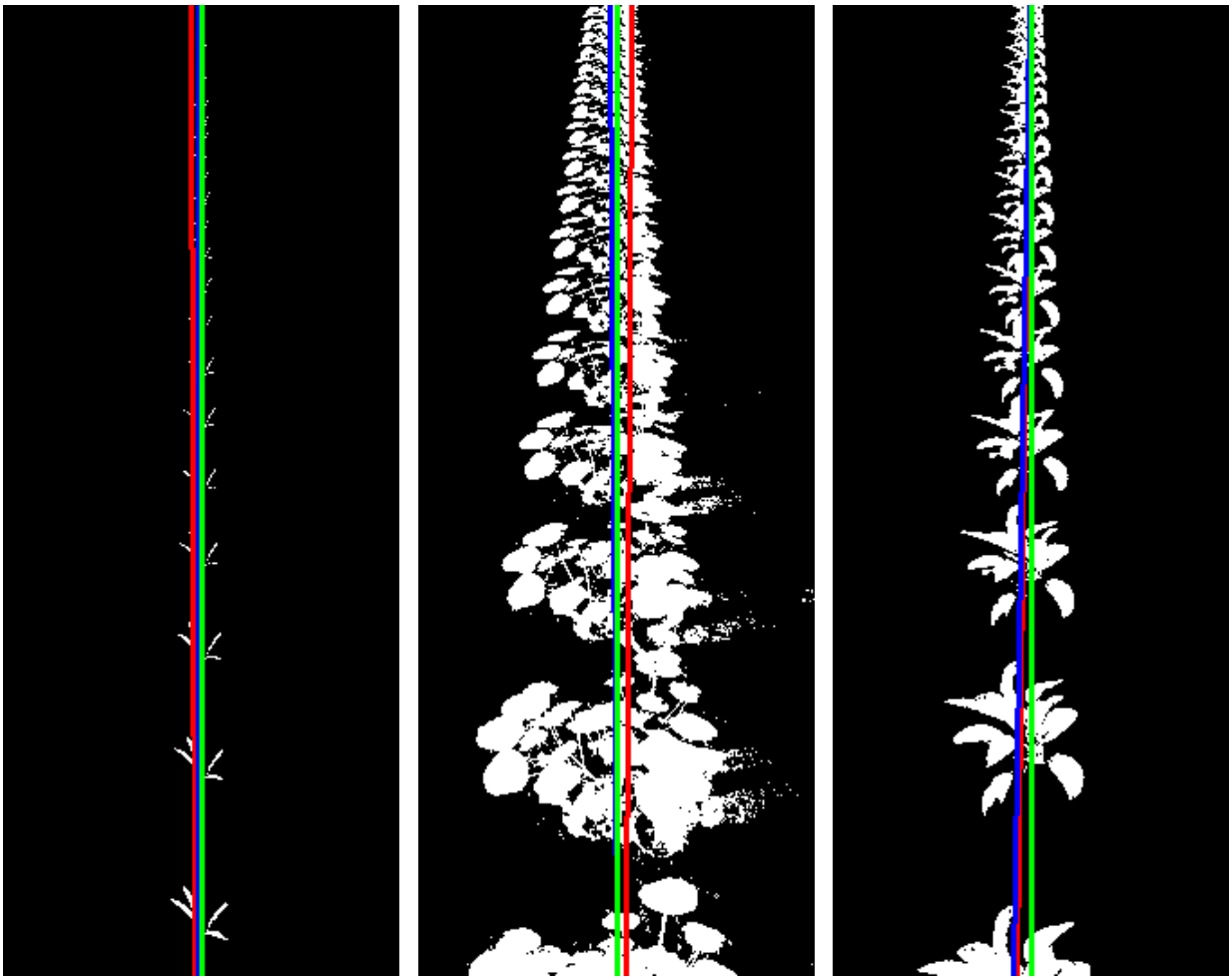


Abbildung 7-6: Analyse der LR (Blau) und SA (Rot), Ideallinie (Grün)

Abbildung 7-7 zeigt den reduzierten Datensatz nachdem der Algorithmus der „Stripe Analysis“ auf das Binärbild angewandt wurde. Wie bereits in Abschnitt 4.5 erläutert, wendet sowohl die „Lineare Regression“ als auch die „Stripe Analysis“ die Methode der kleinsten Fehlerquadrate an, um eine Gerade zwischen den Pflanzen zu approximieren. Der Unterschied der beiden Verfahren liegt lediglich an der Vorselektierung der Daten durch den in Abschnitt 4.5.2 vorgestellten Algorithmus. Aus **Abbildung 7-7** geht hervor, dass auch durch diese Vorselektierung der Ursprung der Pflanze nicht exakt bestimmt werden kann.

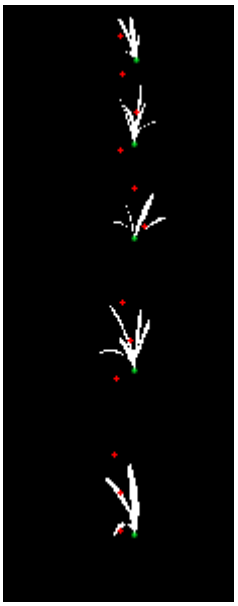


Abbildung 7-7: Kompletter Datensatz (Weiß), reduzierter Datensatz (Rot), „Ground Truth" Daten (Grün)

Abschließend wurden in Abschnitt 7.1.5 noch Messungen mit Unkraut durchgeführt. Dafür wurde die „Blob reduction" angewandt, welche sowohl mit der LR als auch mit der SA gute Ergebnisse liefert. Dass die „Blob reduction" nicht nur Vorteile bringt zeigen allerdings die Ergebnisse aus Abschnitt 7.1.4.2. Grund dafür ist, dass unter gewissen Umständen die Pflanzen nicht mehr als einheitlicher Pixelverbund dargestellt werden (vgl. **Abbildung 7-8**).

Abbildung 7-8: Eingefärbtes Binärbild einer Pflanze (links ohne „Blob reduction", rechts mit)

Die „Blob reduction" würde in so einem Fall die Geometrie der Pflanze grundlegend verändern, was zu negativen Einflüssen auf das Messergebnis führen kann. Aus diesen Gründen wird diese in der vorliegenden Arbeit nur eingesetzt wenn auch Unkraut Teil der Simulation ist.

Der Abschnitt 7.1 hat ganz klar gezeigt, dass sich die „Lineare Regression" besser für die Detektion der Pflanzenreihen eignet. Diese konnte sich bei nahezu allen Simulationen gegenüber der „Stripe Analysis" durchsetzen, weshalb alle nachfolgenden Messungen in dieser Arbeit mit dieser Methode durchgeführt werden.

7.2 Kameraeffekte

Diese Messreihen sollen die Auswirkungen von Kameraeffekten auf den Algorithmus evaluieren. **Abbildung 7-9** zeigt die virtuelle Umgebung ohne Kameraeffekte; dieses Bild dient als Referenz für den gesamte Abschnitt 7.2.

7.2.1 Messreihe Bewegungsunschärfe

7.2.1.1 Beschreibung

Bewegungsunschärfe tritt auf bei relativ rascher Bewegung und langer Belichtungszeit; daher untersucht diese Messreihe die Auswirkungen der Bewegungsunschärfe auf den Algorithmus. Dafür wird der „Motion Blur Amount“ in der „Unreal Engine“ schrittweise von 0% auf 100% erhöht.

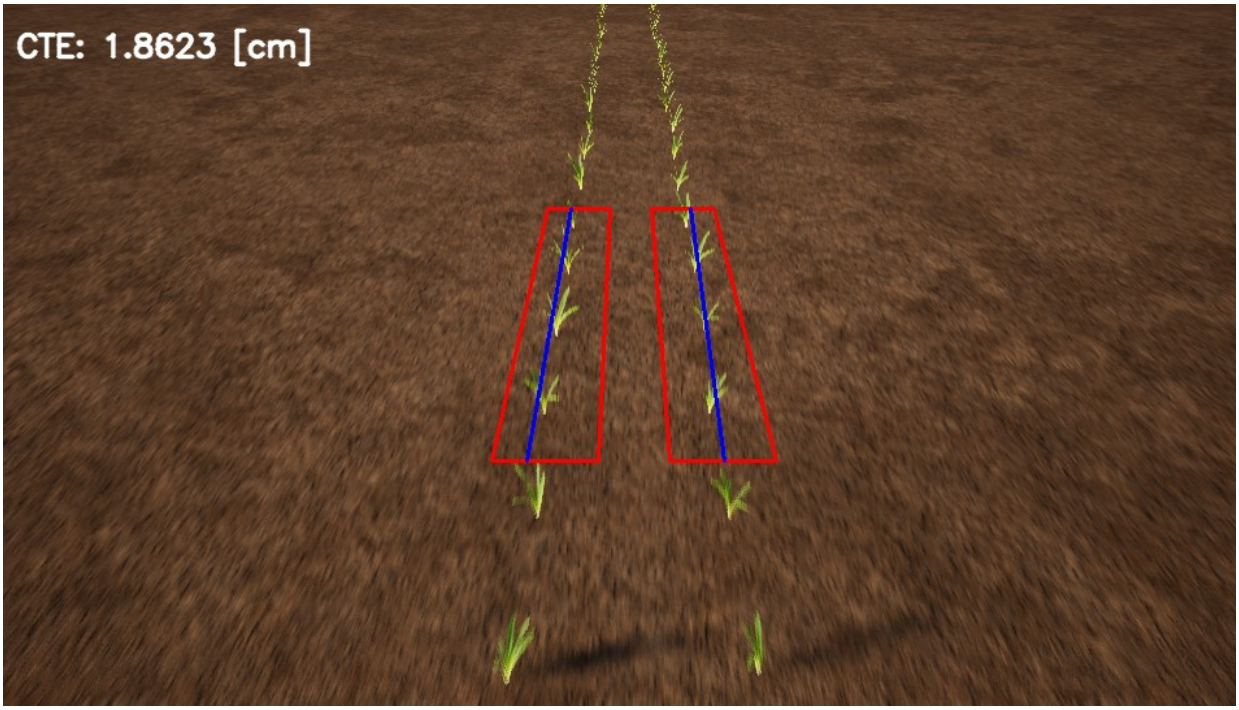


Abbildung 7-9: Virtuelle Umgebung ohne Kameraeffekte

Abbildung 7-10 zeigt die virtuelle Umgebung mit maximaler Bewegungsunschärfe. Wie deutlich zu erkennen ist, ist die Bewegungsunschärfe (engl. „Motion Blur“) im unteren Bildbereich deutlich stärker ausgeprägt als im oberen. Das liegt daran, dass Strukturen die näher an der Kamera liegen eine größere relative Bewegung erfahren.

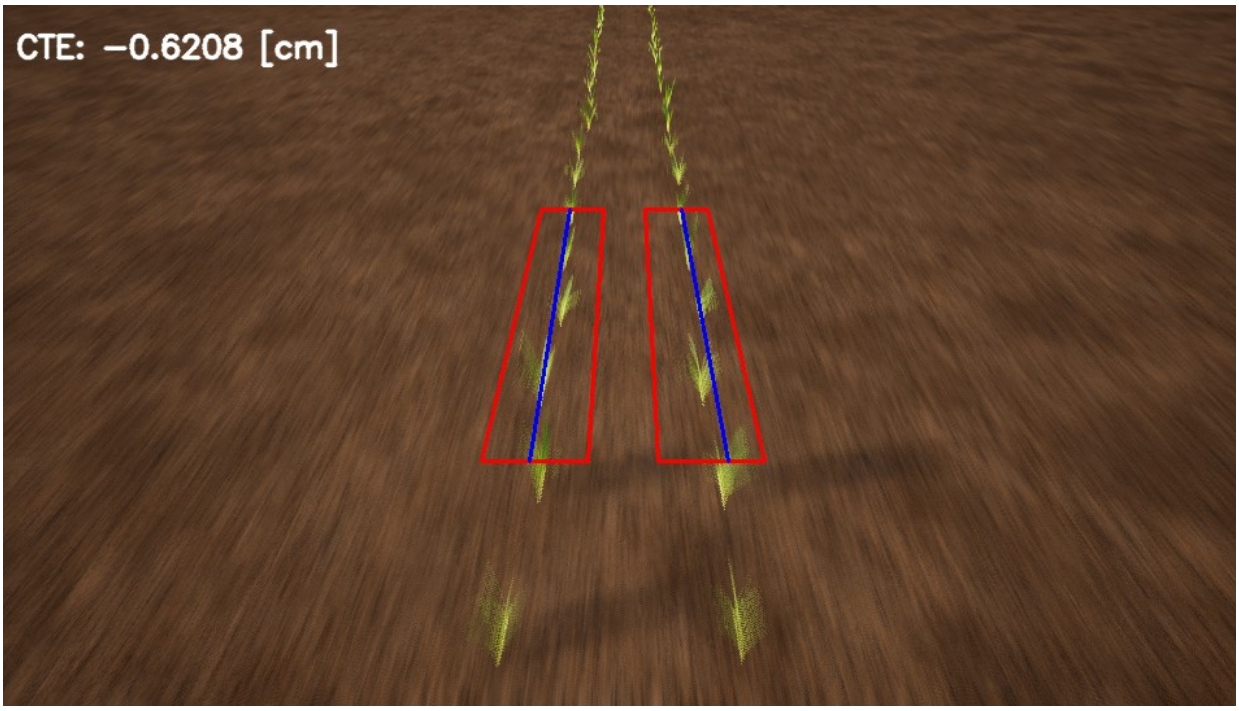


Abbildung 7-10: Virtuelle Umgebung mit einem „Motion Blur Amount“ von 100%

7.2.1.2 Ergebnisse

Abbildung 7-11 stellt das Ergebnis dieser Messreihe als Balkendiagramm graphisch dar. Sowohl Standardabweichung als auch Mittelwert nehmen mit zunehmender Bewegungsunschärfe ab. Betrachtet man die absolute Mittlere Abweichung, so verbessert sich das Messergebnis zwischen den Messungen mit minimaler und maximaler Bewegungsunschärfe um mehr als 20%. Der Grund für den positiven Effekt von Bewegungsunschärfe auf das Messergebnis wird in Abschnitt 7.2.4 erläutert.

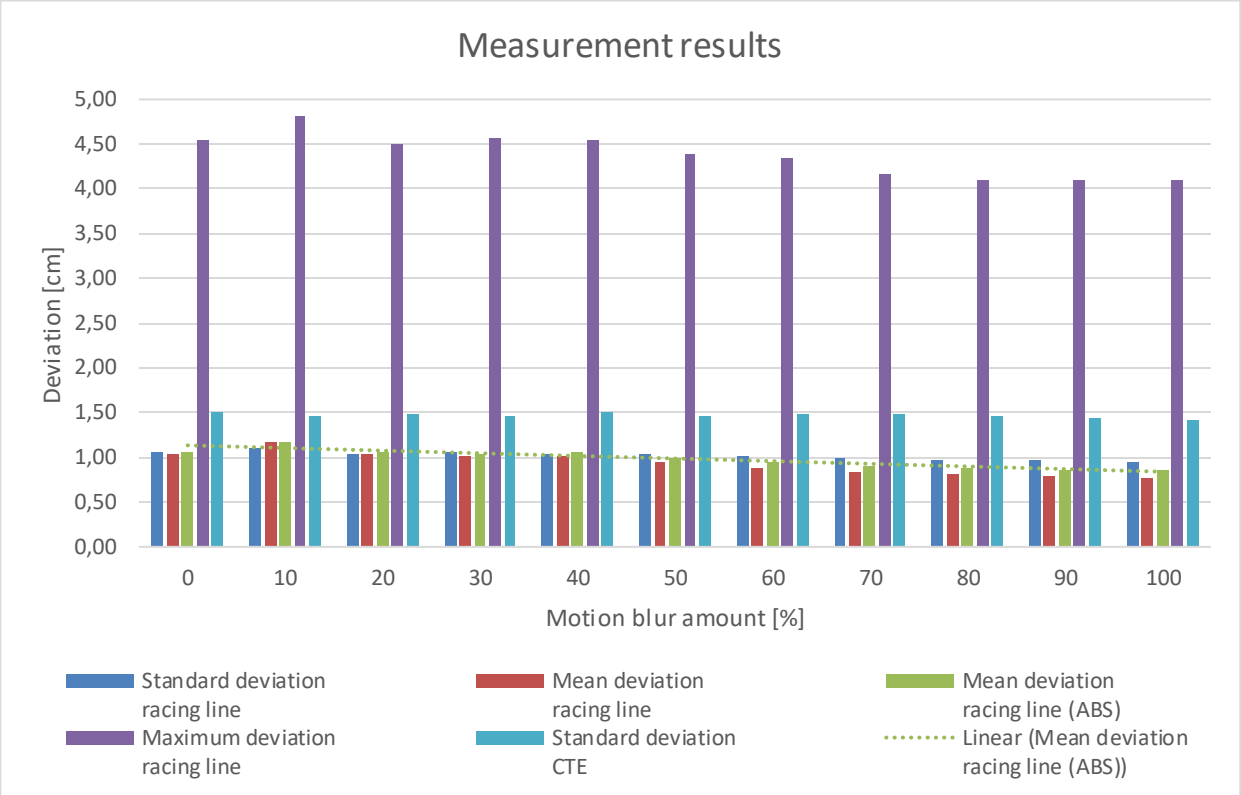


Abbildung 7-11: Messergebnisse Bewegungsunschärfe

7.2.2 Messreihe Bildrauschen

7.2.2.1 Beschreibung

Bildrauschen ist wohl eine der bekanntesten Störungen in der Fotographie. Dieser Abschnitt widmet sich daher der Untersuchung dieses Problems. Bei dieser Messreihe werden der „Grain Jitter“ und die „Grain Intensity“ in der „Unreal Engine“ von 0% auf 100% sukzessive erhöht. Der „Grain Jitter“ spiegelt die Bildstabilität wider und die „Grain Intensity“ die Intensität des Rauschens.

7.2.2.2 Ergebnisse

Abbildung 7-12 zeigt die Messergebnisse der Messreihe Bildrauschen. Ähnlich wie in Abschnitt 7.1.5, nehmen die Standardabweichung und der mittlere Fehler mit zunehmendem Bildrauschen ab. Die Abweichung zwischen erster und letzter Messung ist zwar nicht mehr so drastisch wie in Abschnitt 7.1.5, doch wird das Messergebnis immerhin noch um knapp 15% besser.

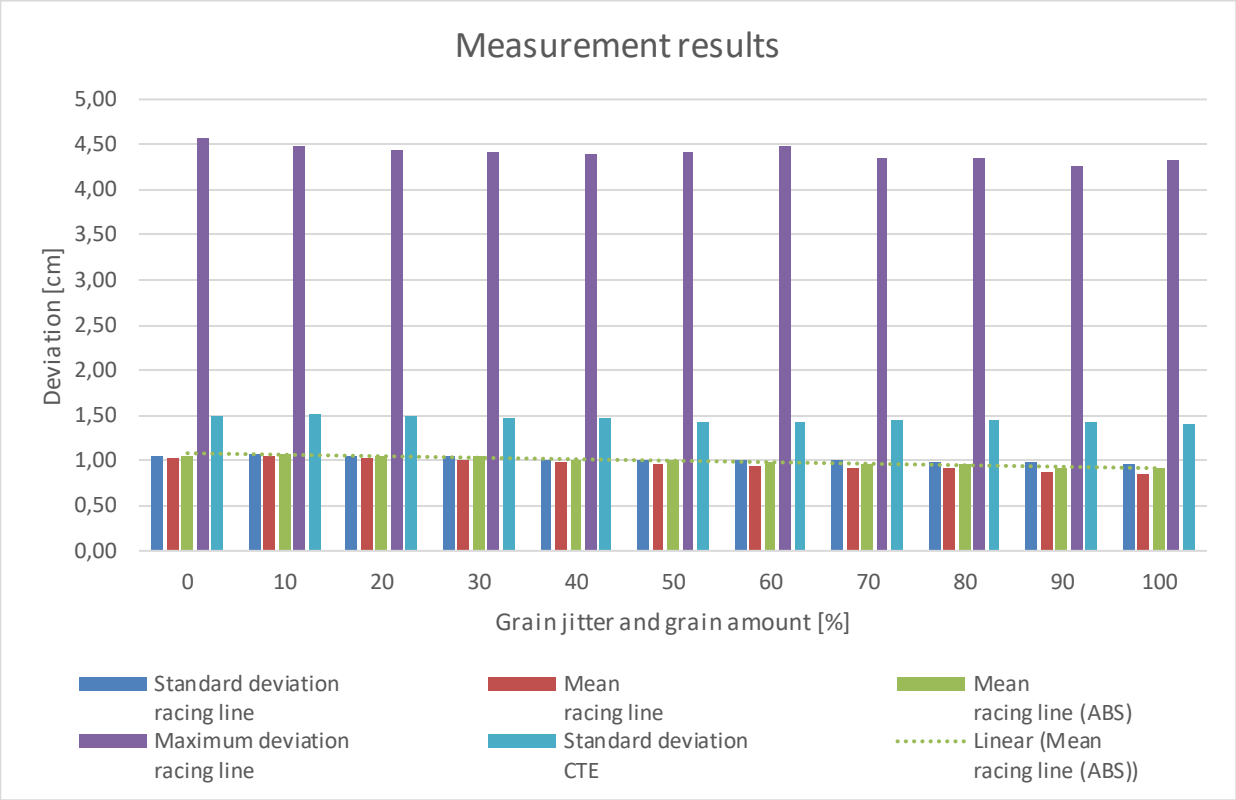


Abbildung 7-12: Messergebnisse Bildrauschen

7.2.3 Messreihe Linsenverschmutzung

7.2.3.1 Beschreibung

Der Schmutzmasken-Effekt (engl. „Dirt Mask effect“) der „Unreal Engine“ verwendet eine Textur, um den „Blooming Effekt“ in definierten Bildbereichen aufzuhellen [56]. „Blooming“ kommt bei CCD-Sensoren vor, falls die Speicherkapazität eines lichtempfindlichen Elements überschritten und die Ladungsmenge dadurch an das benachbarte Element übertragen wird [57, p. 176]. Obwohl die vorliegende Arbeit auf einem Kamerasystem mit CMOS-Sensor beruht, soll der Schmutzmasken-Effekt in dieser Messreihe genutzt werden, um Linsenverschmutzungen der Kamera zu simulieren. Die verwendeten Schmutzmasken ist die „T_Metal_Rust_D“ welche eine Verschmutzung der Linse darstellet, wie sie beispielsweise durch aufgewirbelten Sand entstehen könnte. **Abbildung 7-13** zeigen das resultierende Bild bei maximaler Verschmutzung.

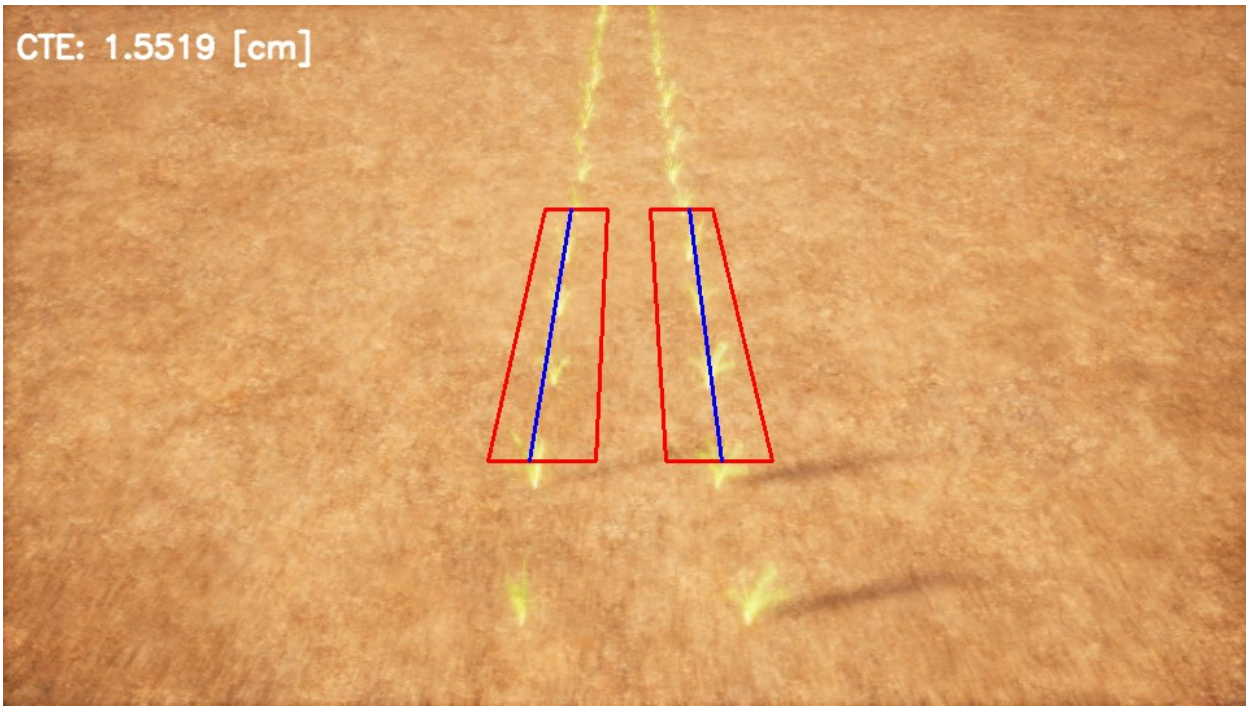


Abbildung 7-13: Virtuelle Umgebung Textur: „T_Metal_Rust_D“ (100%)

7.2.3.2 Ergebnisse

Die Simulationsergebnisse der Messreihe Linsenverschmutzung werden in **Tabelle 7-5** dargestellt. Vergleicht man die Ergebnisse dieser Messreihe mit jenen aus den Abschnitten 7.2.1 und 7.2.2. so stellt man fest, dass diese nahezu ident sind. Die Abnahme zwischen erster und letzter Messung ist mit gut 20% gleich groß wie auch bei der Simulation der Bewegungsunschärfe.

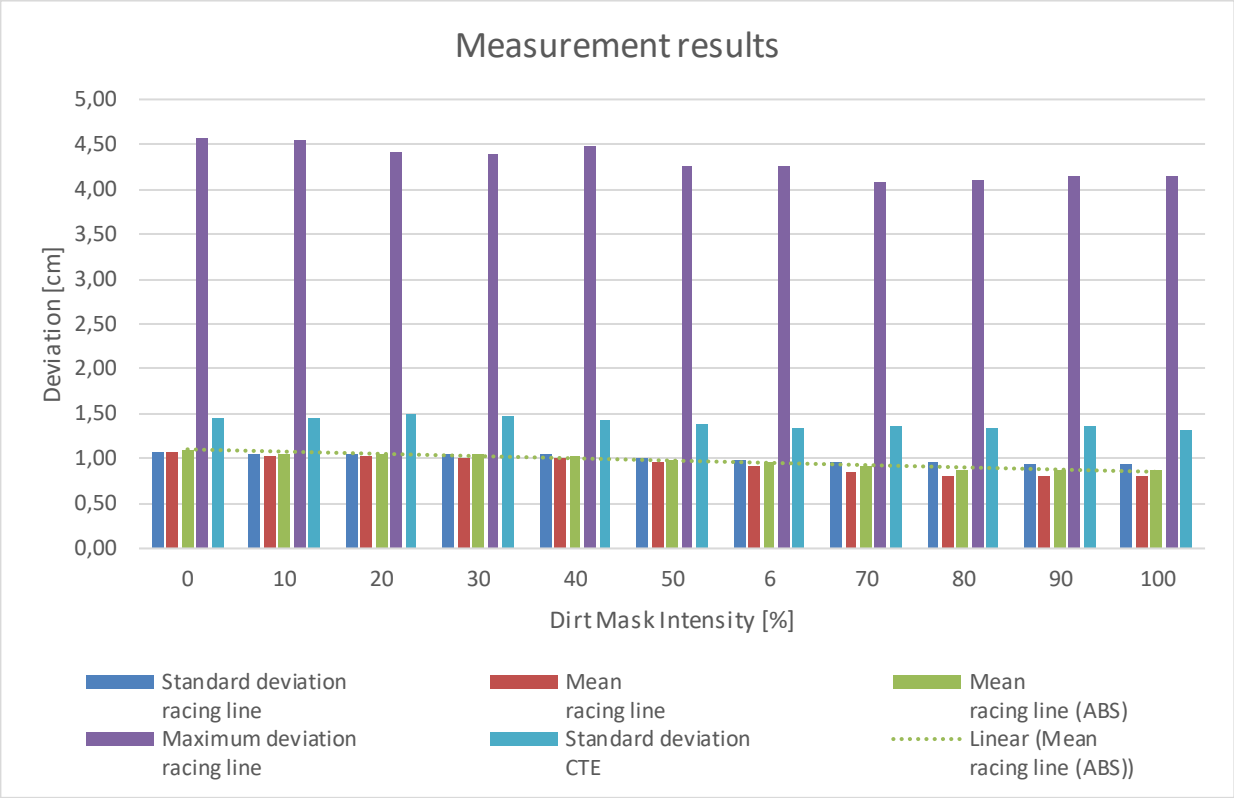


Tabelle 7-5: Messergebnis Linsenverschmutzung

7.2.4 Analyse der Kameraeffekte

Aufgrund der nahezu identen Messergebnisse aus den Abschnitten 7.2.1 (Bewegungsunschärfe), 7.2.2 (Bildrauschen) und 7.2.3 (Linsenverschmutzung) ist darauf zu schließen, dass es einen einheitlichen Grund für die Verbesserung der Messergebnisse mit zunehmenden Kameraeffekten gibt. Es wird vermutet, dass sich die Messergebnisse aufgrund von zunehmend homogeneren Pflanzen-Geometrien verbessern. Diese Annahme wird in diesem Abschnitt mit dem „Digital Plant Analyzer“ (DPA) bestätigt. Dieser wurde an der Universität Hohenheim, Deutschlands Nr. 1 in der Agrarforschung [58], entwickelt. Mithilfe dieser Software lassen sich der „Excess green index“ (ExG), der „Excess red index“ (ExR), der „ExG minus ExR vegetation index“ (ExGR) sowie eine Farbfilterfunktion mit den Farbräumen HSV und CieLab untersuchen. Der Schwellwert für den ExG, den ExR und den ExGR kann entweder über die Otsu Schwellwertmethode oder manuell vorgegeben werden. Der DPA bietet die Möglichkeit, das Originalbild in Graustufen oder als Binärbild darzustellen und abzuspeichern. Weiters kann damit das Histogramm des HSV und des CieLab-Kanals angezeigt werden [59]. Der Quellcode für diese Software kann unter https://github.com/hohenheimdr/DPA heruntergeladen werden.

Abbildung 7-14 zeigt das Binärbild aus Abschnitt 7.2.1.1. Wie erwähnt, ist die Bewegungsunschärfe im unteren Teil der Abbildung aufgrund der Relativbewegung deutlich stärker ausgeprägt. Die Geometrie der Pflanzen „verschwimmt“, was dazu führt, dass der

Algorithmus den Schwerpunkt der Pflanzen besser einschätzt und daher bessere Ergebnisse bei starker Bewegungsunschärfe liefert.

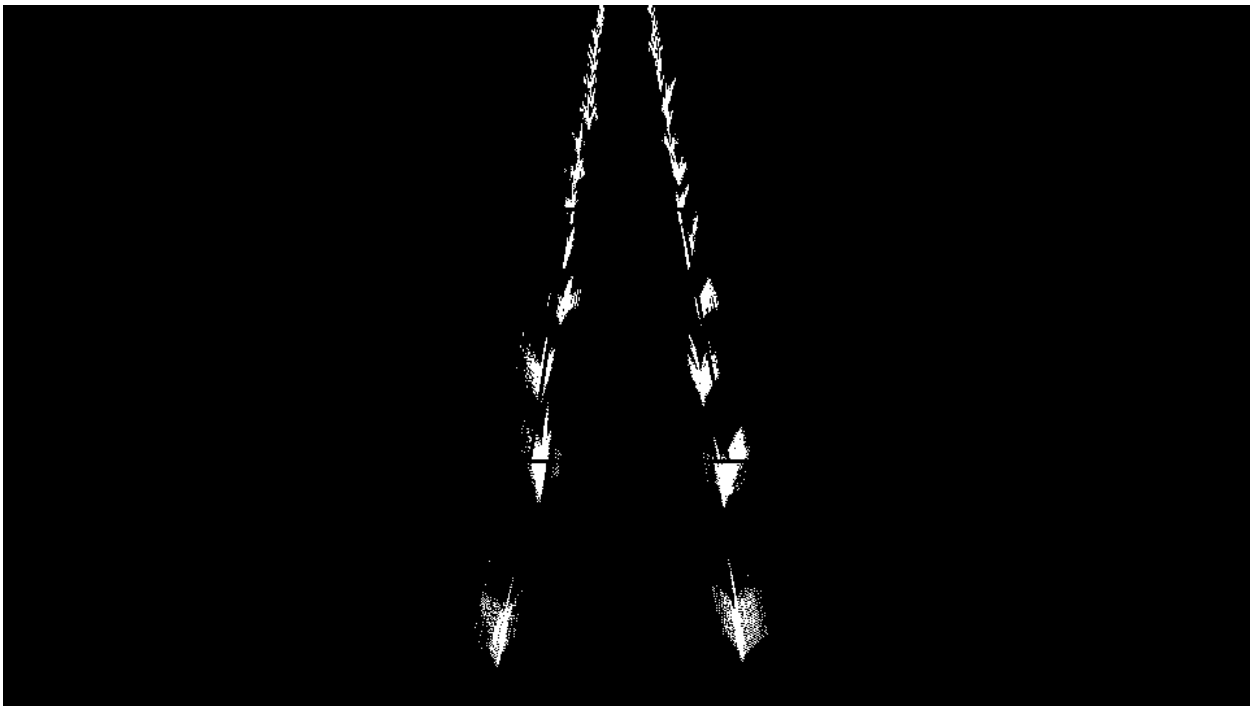


Abbildung 7-14: Binärbild generiert mit Otsu Schwellwertmethode aus Abbildung 7-10

Noch deutlicher ist dieser Effekt bei der Linsenverschmutzung zu erkennen. Wie bereits in Abschnitt 7.2.3.1 beschrieben, verwendet die „Unreal Engine“ hier eine Textur, um den „Blooming Effekt“ in definierten Bildschirmbereichen aufzuhellen. Da dieser Effekt auf die ganze Szene angewandt wird, führt die Otsu Schwellwertmethode lediglich bei einer „Region of Interest“ zum gewünschten Ergebnis. Zwecks besserer Darstellung wurde der Schwellwert in **Abbildung 7-15** manuell festgelegt. Der Effekt der „verloren gegangenen Geometrie“ der Pflanzen ist hier gut zu erkennen.

Dieser Effekt ist für den Abschnitt 7.2.2 Messreihe Bildrauschen nicht nachweisbar, allerdings nimmt die Genauigkeit des Algorithmus mit zunehmendem Rauschen auch nicht so stark zu wie bei der Bewegungsunschärfe und der Linsenverschmutzung.

Mit einer durchschnittlichen absoluten Messabweichung von rund 1 cm und einer maximalen Abweichung von circa 5 cm stellen Kameraeffekte kein Problem für den Algorithmus dar.

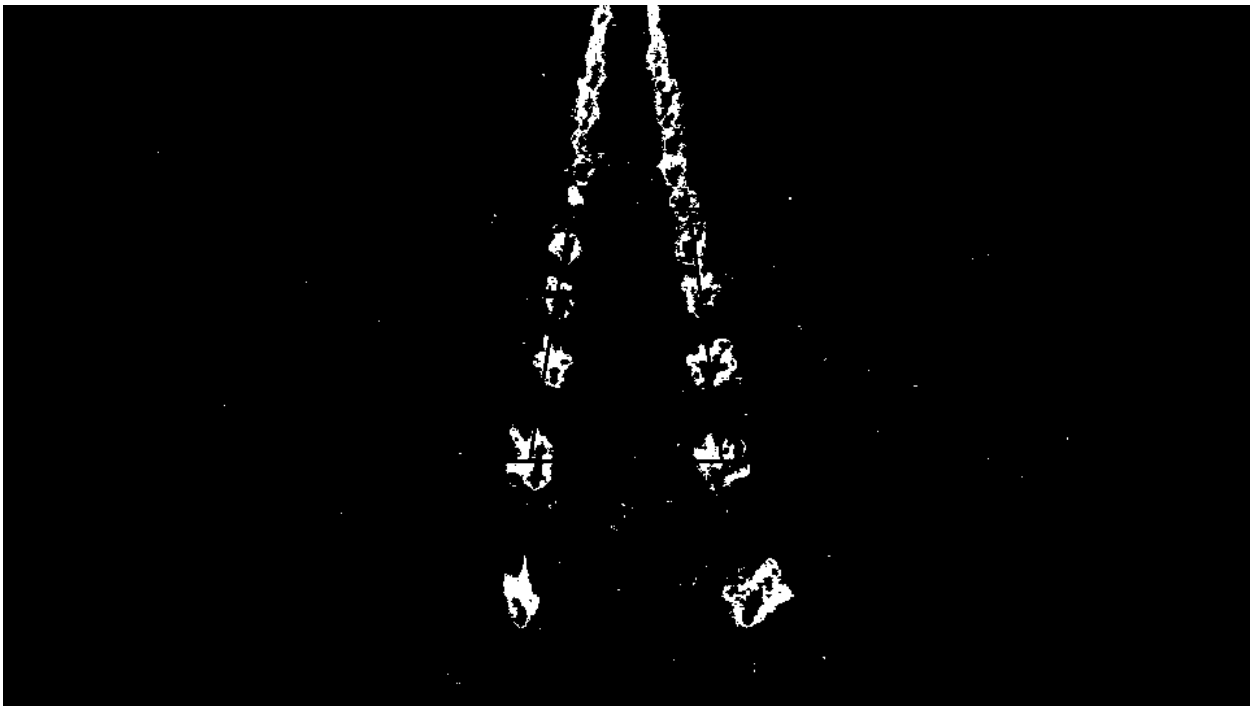


Abbildung 7-15: Binärbild manueller Schwellwert 20, generiert aus Abbildung 7-13

Abschließend sei erwähnt, dass unerwünschte Kameraeffekte einen überraschend positiven Einfluss auf die Genauigkeit des Algorithmus haben. Das verwendete Kameramodell, mit allen seinen Einschränkungen (vgl. Kapitel 6), ist vollkommen ausreichend, um den zu untersuchenden Algorithmus zu evaluieren.

7.3 Robustheit gegenüber Farbabweichungen

Um eine korrekte Unterscheidung zwischen Pflanzen und Boden treffen zu können, ist ein präziser Vegetationsindex erforderlich [13]. Der vorliegende Algorithmus setzt hier auf den „Excess green index“ (ExG) [1, p. 26]. Die folgenden Abschnitte wollen daher die Frage klären, wie gut sich dieser Index zur Detektion von Pflanzenreihen eignet. Zu diesem Zweck werden mehrere Messreihen durchgeführt, bei denen neben der Farbtemperatur des gerichteten Lichts auch die Sonnenstände variiert werden.

7.3.1 Messreihen Variation der Farbtemperatur

7.3.1.1 Beschreibung

Mithilfe dieser Messreihen soll untersucht werden, wie robust der Algorithmus gegenüber Farbabweichungen ist. Dazu werden Messungen mit unterschiedlichen Böden und Pflanzen durchgeführt und die Farbtemperatur des Lichts variiert. Insgesamt werden in diesem Abschnitt vier Messreihen durchgeführt, die **Tabelle 7-6** auflistet.

Messreihe	Bezeichnung
1	Boden 1 - Pflanze 1
2	Boden 1 - Pflanze 2
3	Boden 2 - Pflanze 1
4	Boden 2 - Pflanze 2

Tabelle 7-6: Auflistung der durchgeführten Messreihen

7.3.1.2 Ergebnisse

Abbildung 7-16 stellt nicht nur die Messergebnisse der 1. Messreihe dar, sondern präsentiert unter anderem eine typische Charakteristik aller vier Messreihen. Die einzelnen Ergebnisse ändern sich durch die Variation der Lichtfarbe kaum. Auch die einzelnen statistischen Kenngrößen weisen keine Besonderheiten auf.

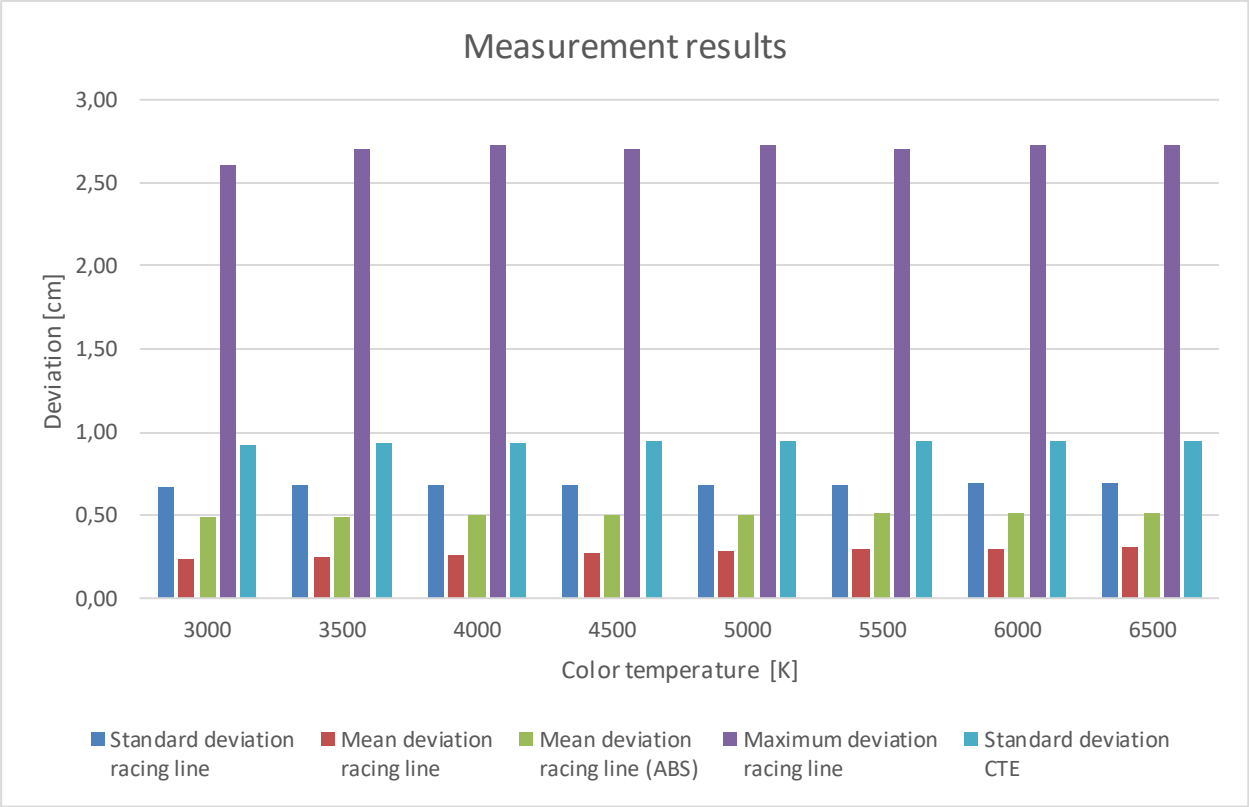


Abbildung 7-16: Ergebnisse Messreihe 1: Boden 1 – Pflanze 1

Die Standardabweichungen des CTE, die bekanntlich indirekt für die Lenkkorrektur verantwortlich sind, sind wie bei den vorangegangenen Messungen größer als die Standardabweichung der Ideallinie. Die Mittlere Abweichung der absoluten Beträge ist teilweise doppelt so groß wie jene der relativen Abweichungen, was darauf hindeutet, dass sich das

Implement im Lauf der Simulation sowohl links als auch rechts der Ideallinie aufhält. Ein identer Wert der „Mean deviation racing line“ und der „Mean deviation racing line“ (ABS) würde bedeuten, dass sich das Implement stets auf einer Seite der Ideallinie befindet, was auf ein ungenaues Ergebnis verweisen würde (vgl. Abschnitt 4.7).

Aufgrund der Tatsache, dass alle vier Messergebnisse eine sehr ähnliche Charakteristik aufweisen, wird auf deren graphische Darstellung verzichtet. Stattdessen werden die Ergebnisse in tabellarischer Form in **Tabelle 7-7** und **Tabelle 7-8** dargestellt.

Measurement Number	Name	Value	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE	
			P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
1	Color Temp.	3000K	0,67	0,69	0,19	0,49	0,47	0,59	2,70	2,79	0,92	0,75
2	Color Temp.	3500K	0,68	0,69	0,26	0,48	0,49	0,58	2,72	2,79	0,92	0,75
3	Color Temp.	4000K	0,69	0,68	0,28	0,47	0,50	0,58	2,76	2,76	0,93	0,75
4	Color Temp.	4500K	0,69	0,68	0,30	0,47	0,51	0,57	2,79	2,76	0,93	0,76
5	Color Temp.	5000K	0,69	0,69	0,32	0,46	0,52	0,57	2,82	2,76	0,93	0,76
6	Color Temp.	5500K	0,69	0,68	0,34	0,45	0,52	0,56	2,86	2,73	0,93	0,76
7	Color Temp.	6000K	0,70	0,68	0,35	0,45	0,53	0,56	2,89	2,79	0,94	0,76
8	Color Temp.	6500K	0,70	0,68	0,37	0,44	0,54	0,56	2,89	2,79	0,92	0,76

Tabelle 7-7: Vergleich der Messergebnisse mit Pflanze 1 (P1) und Pflanze 2 (P2) und jeweils Boden 2

Aus **Tabelle 7-7** lässt sich erkennen, dass sich lediglich die mittleren Abweichungen signifikant ändern. Das ist auf die Geometrie der Pflanzen zurückzuführen, da der Algorithmus den Schwerpunkt der Pflanzen falsch einschätzt. Eine weitere Analyse der Rohdaten, die mittels „Linearer Regression“ erfolgte, hat ergeben, dass sich das Implement im Lauf der Simulation tendenziell zu weit links von der Ideallinie aufhält, wenn mit der Pflanze 2 simuliert wird.

Measurement Number	Name	Value	Standard deviation racing line		Mean deviation racing line		Mean deviation racing line (ABS)		Maximum deviation racing line		Standard deviation CTE	
			S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
1	Color Temp.	3000K	0,68	0,69	0,49	0,49	0,59	0,59	2,73	2,79	0,70	0,75
2	Color Temp.	3500K	0,68	0,69	0,49	0,48	0,58	0,58	2,70	2,79	0,71	0,75
3	Color Temp.	4000K	0,68	0,68	0,49	0,47	0,58	0,58	2,76	2,76	0,71	0,75
4	Color Temp.	4500K	0,68	0,68	0,49	0,47	0,58	0,57	2,76	2,76	0,71	0,76
5	Color Temp.	5000K	0,68	0,69	0,48	0,46	0,58	0,57	2,79	2,76	0,75	0,76
6	Color Temp.	5500K	0,68	0,68	0,48	0,45	0,58	0,56	2,76	2,73	0,70	0,76
7	Color Temp.	6000K	0,68	0,68	0,48	0,45	0,58	0,56	2,76	2,79	0,70	0,76
8	Color Temp.	6500K	0,68	0,68	0,46	0,44	0,57	0,56	2,76	2,79	0,71	0,76

Tabelle 7-8: Vergleich der Messergebnisse mit Boden 1 (S1) und Boden 2 (S2) und jeweils Pflanze 2

Die Ergebnisse der Messreihen 2 und 4 finden sich in **Tabelle 7-8**. Diese stützt die oben genannte These über die Geometrie der Pflanzen. Wie ersichtlich, haben sich die Änderungen der mittleren Abweichungen auf ein Minimum reduziert. Weiters lässt sich aus diesen Messdaten schließen, dass das Bodenmaterial keinen signifikanten Einfluss auf die Messergebnisse hat.

7.3.2 Messreihe Farbsättigung der Pflanzen

7.3.2.1 Beschreibung

Bei dieser Messreihe wird das „Substance Plugin“ verwendet. Dieses Plugin bietet sogenannte „Substance materials“, bei denen es sich um anpassbare Materialien handelt, die von Spezialisten und erstklassigen Gastkünstlern erstellt wurden. Die modifizierbaren Parameter ermöglichen unzählige Variationen [60]. Für unser Material mit dem Namen „farm_material“ bedeutet das drei generische Pflanzentypen, veränderbare Parameter wie Pflanzenfarbe, Skalierung, Bodenunebenheit, Bodenverformung, Bodenfeuchtigkeitsmenge, Steine, Zweige und Wasserstände [61]. Unter anderem besteht die Möglichkeit, die Farbsättigung der Pflanzen zwischen 0% und 100% zu variieren (vgl. **Abbildung 7-17**). Das ist allerdings nur möglich, wenn eine der drei generischen Pflanzentypen des Plugins verwendet wird. Der Nachteil daran ist, dass es nicht möglich ist, die Positionen dieser Pflanzen auszulesen. Ohne diese Information kann auch die Ideallinie nicht korrekt berechnet werden, weshalb in **Abbildung 7-18** nur der „Cross Track

Error“ zur Bewertung herangezogen wird. Zwar wird in den Rohdaten auch der Mittelwert des CTE berechnet; da dieser jedoch bei sämtlichen Messungen annähernd Null ist, wird auf seine Darstellung in den Diagrammen verzichtet.

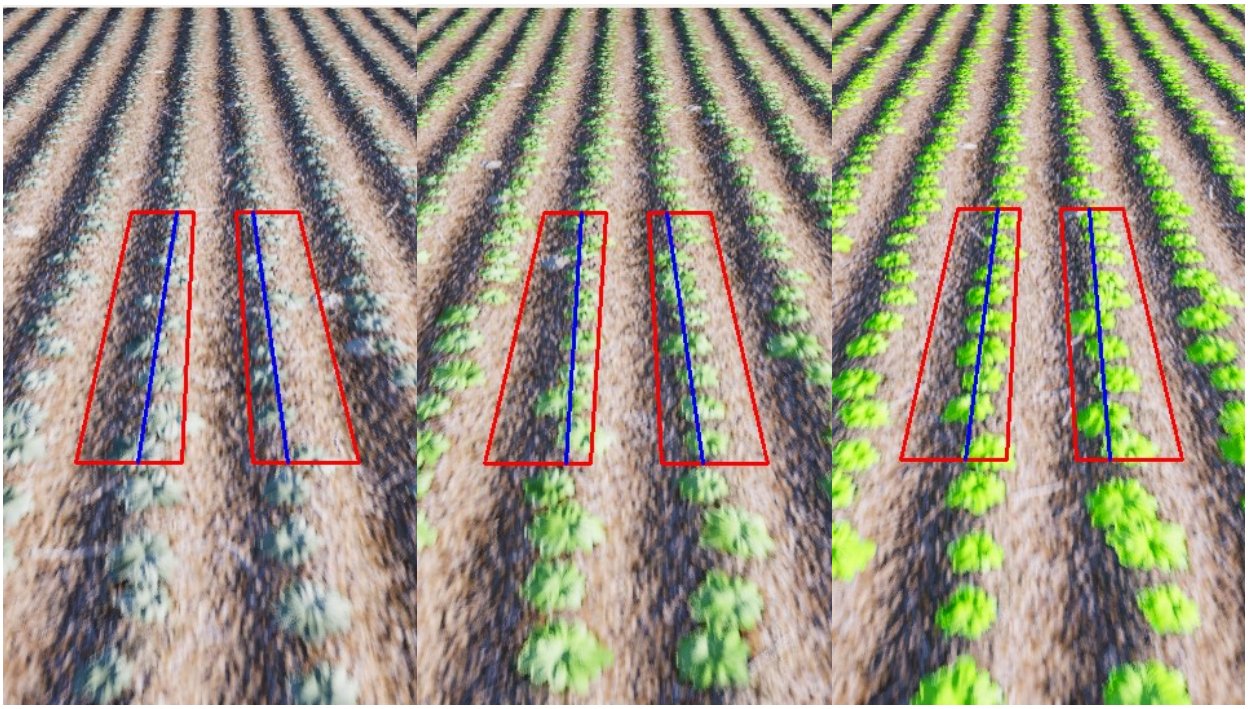


Abbildung 7-17: Farbsättigung der Pflanzen von links nach rechts, saturation: 42%, 50%, 100%

7.3.2.2 Ergebnisse

Abbildung 7-18 zeigt das Messergebnis der aktuellen Messreihe. Messungen, bei denen die Farbsättigung der Pflanzen unter 42% liegt, sind nicht dargestellt, da der Algorithmus nicht mehr in der Lage ist, die Reihen ordnungsgemäß zu detektieren und der CTE daher exorbitant steigt. Wie in der **Abbildung 7-18** zu sehen ist, entspricht eine Farbsättigung von 42% aber nicht mehr der Realität.

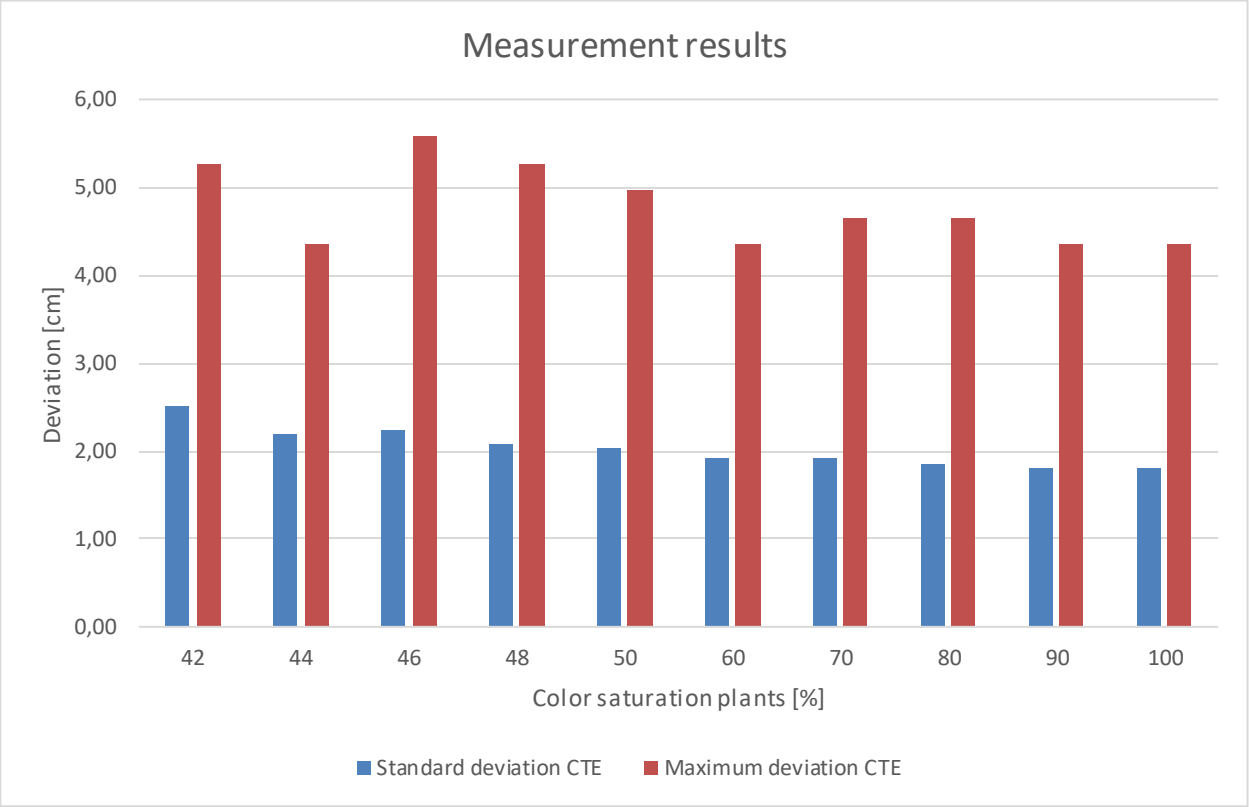


Abbildung 7-18: Messergebnisse Farbsättigung der Pflanze

7.3.3 Messreihe zusätzliche Grünflächen

7.3.3.1 Beschreibung

Auf einer realen Ackerfläche befinden sich neben der eigentlichen Nutzpflanze auch Gras und Unkraut. Mithilfe des „Substance Plugin“ soll festgestellt werden, ob zusätzliche Grasflächen Auswirkungen auf die Pflanzenreihenerkennung haben. Wie bereits in Abschnitt 7.3.2 ist die Position der Pflanzen unbekannt, weshalb die Auswertung in **Abbildung 7-20** wieder auf dem CTE beruht. Für die Messreihe wird der „Ground grass amount“ von 0% auf 20% erhöht. **Abbildung 7-19** zeigt die virtuelle Umgebung mit einem „Ground grass amount“ von 20%.

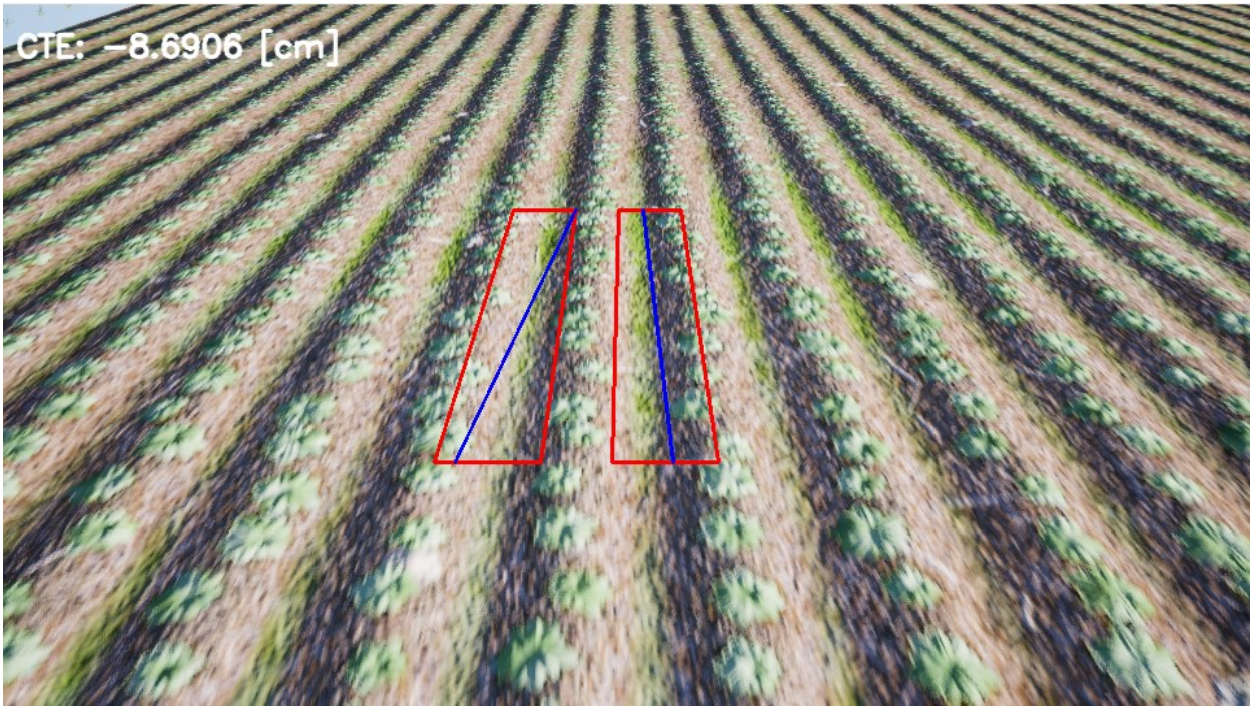


Abbildung 7-19: Virtuelle Umgebung mit zusätzlichen Grasflächen; „Ground grass amount“: 20%

7.3.3.2 Ergebnisse

Der Algorithmus reagiert im ersten Teil der Messreihe kaum auf die zusätzlichen Grasflächen. Ab 16% zusätzlicher Grasfläche ist in **Abbildung 7-20** erstmals eine Veränderung im Messergebnis zu verzeichnen. Erhöht man die Grasflächen danach jedoch noch weiter, ist eine rapide Verschlechterung zu erkennen. Eine Erhöhung um lediglich 2% führt zu einer Verdoppelung der maximalen Abweichung. Erhöht man die Grasflächen um weitere 2%, ist der Algorithmus nicht mehr imstande, die Spur über die komplette Simulationsdauer hinweg zu halten (vgl. **Abbildung 7-19**).

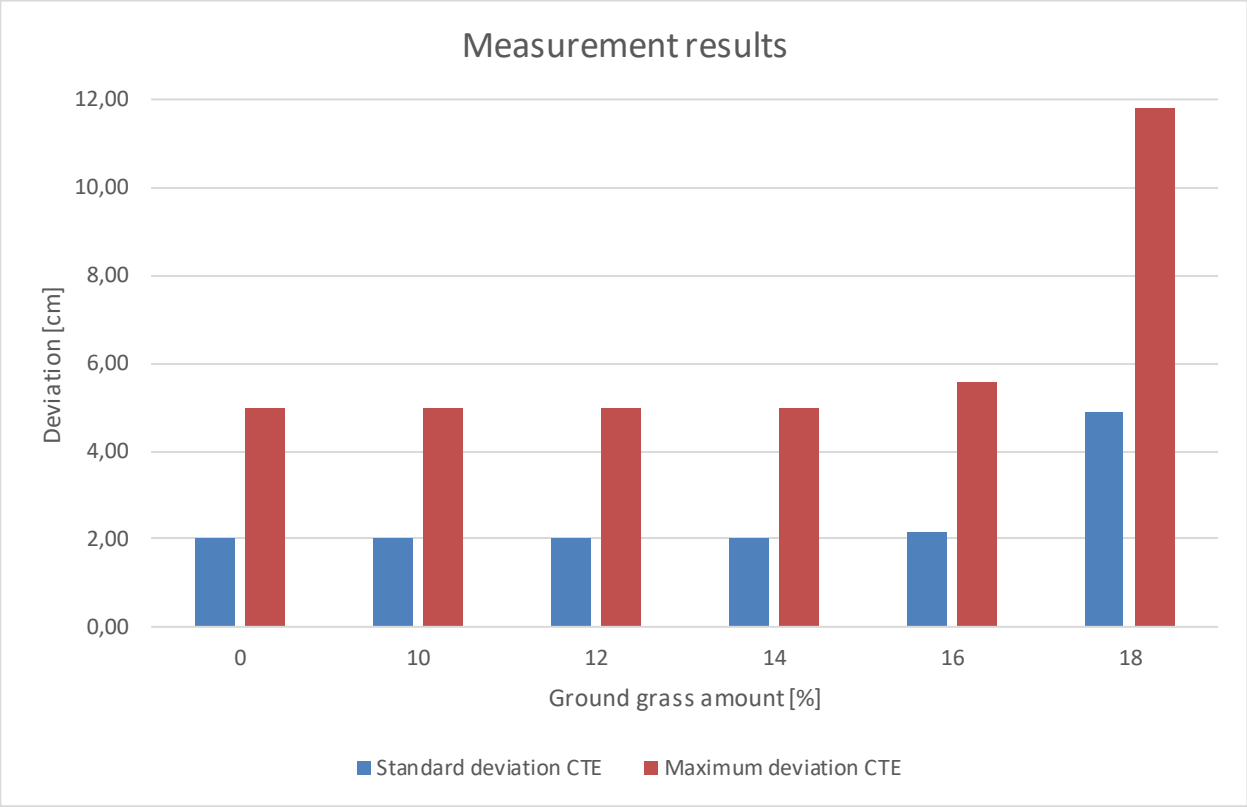


Abbildung 7-20: Messergebnisse zusätzliche Grasflächen

7.3.4 Messreihen Sonnenstände

7.3.4.1 Beschreibung

Die folgenden Messreihen sollen klären, ob die Tageszeit einen Einfluss auf die Genauigkeit des Algorithmus hat. Zu diesem Zweck wurden zwischen Anfang Mai und Mitte Juni 4 Messreihen simuliert. Die Sonnenstände wurden dabei mittels des „Sun Position Calculators“ (vgl. Abschnitt 5.3.2.1) zwischen 6 Uhr morgens und 20 Uhr abends simuliert. **Abbildung 7-21** zeigt die unterschiedlichen Lichtverhältnisse im Lauf eines Tages. Um das Messergebnis für die folgende Analyse nicht unnötig zu beeinflussen, wird in diesem Abschnitt auf die Rotation der Pflanzen verzichtet.

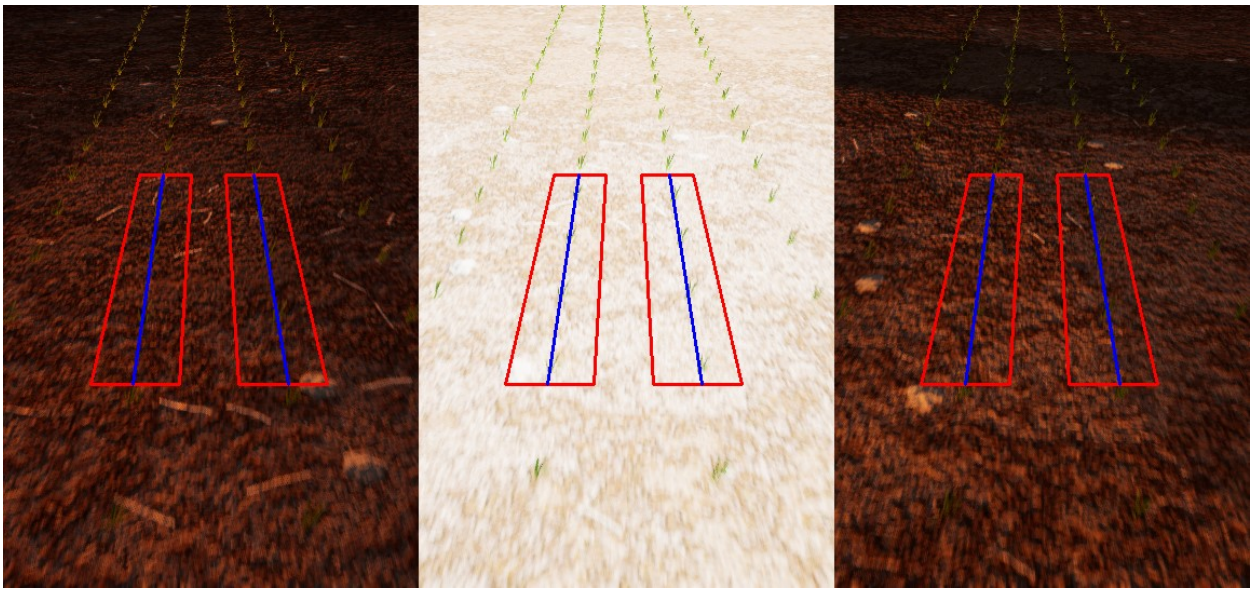
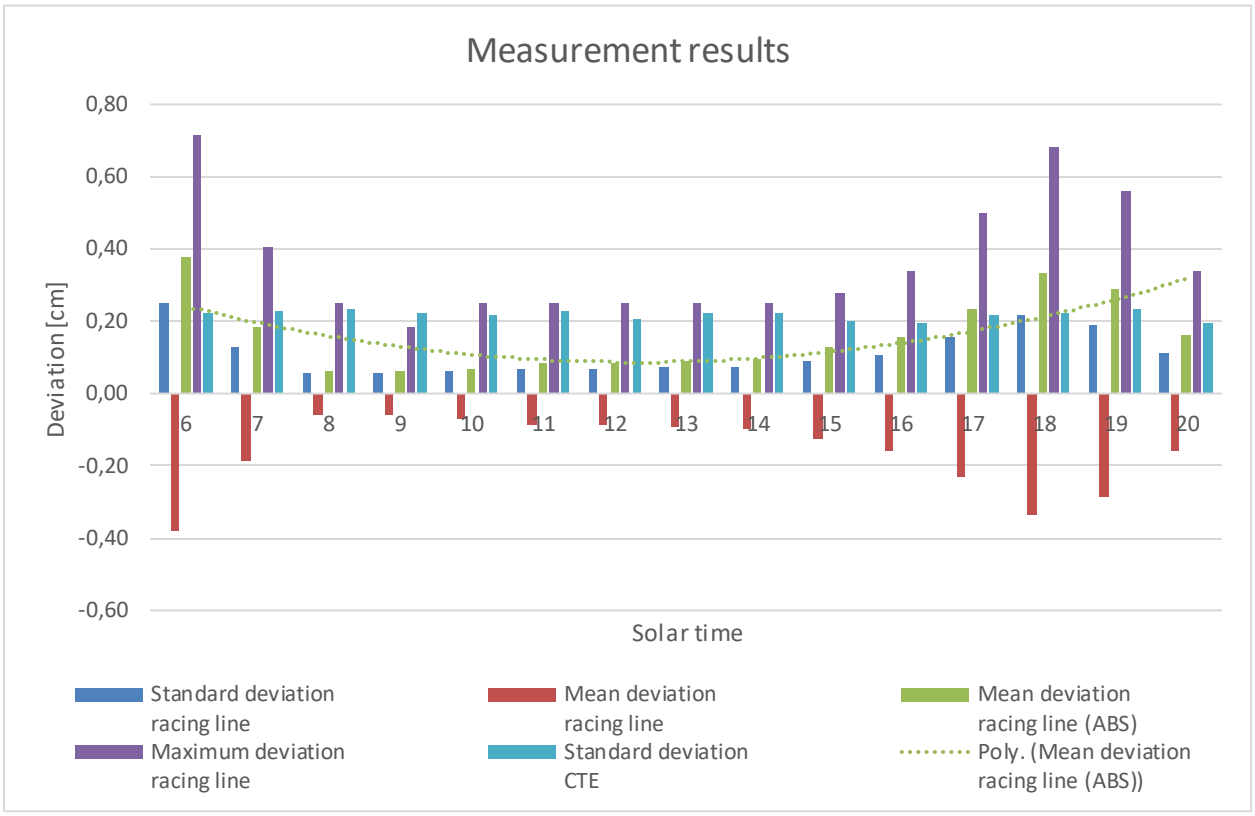


Abbildung 7-21: Virtuelle Umgebung am 1. Mai zu unterschiedlichen Tageszeiten (von rechts nach links solar time: 6 Uhr, 12 Uhr und 20 Uhr)

7.3.4.2 Ergebnisse

Es wurden 4 Messreihen zu je 15 Messungen durchgeführt (siehe **Abbildung 7-22** und **Abbildung 7-23**). Bis auf jene Messreihe, die den 1. Mai simuliert, weisen alle anderen eine sehr ähnliche Charakteristik auf. Die größten Abweichungen findet man morgens um 6 und abends um 20 Uhr. Bis zur Mittagszeit reduziert sich der mittlere absolute Fehler teilweise auf ein Fünftel. Dabei sollte man jedoch nicht aus den Augen verlieren, dass es sich bei dieser Reduktion lediglich um eine Verbesserung um etwa 0,4 cm handelt. Das negative Vorzeichen der relativen mittleren Abweichung ist typisch für Messreihen, bei denen die Pflanzen weder rotiert noch mit einem Offset versehen werden. Dass liegt daran, dass die im Laufe der Simulation gefahrene Linie größtenteils rechts von der Ideallinie liegt (vgl. Abschnitt 7.4.4.2). Der nicht vorhandene Pflanzenoffset führt nebenbei dazu, dass auch der „Cross Track Error“ und die maximale Abweichung im Vergleich zu anderen Messungen deutlich geringer sind.



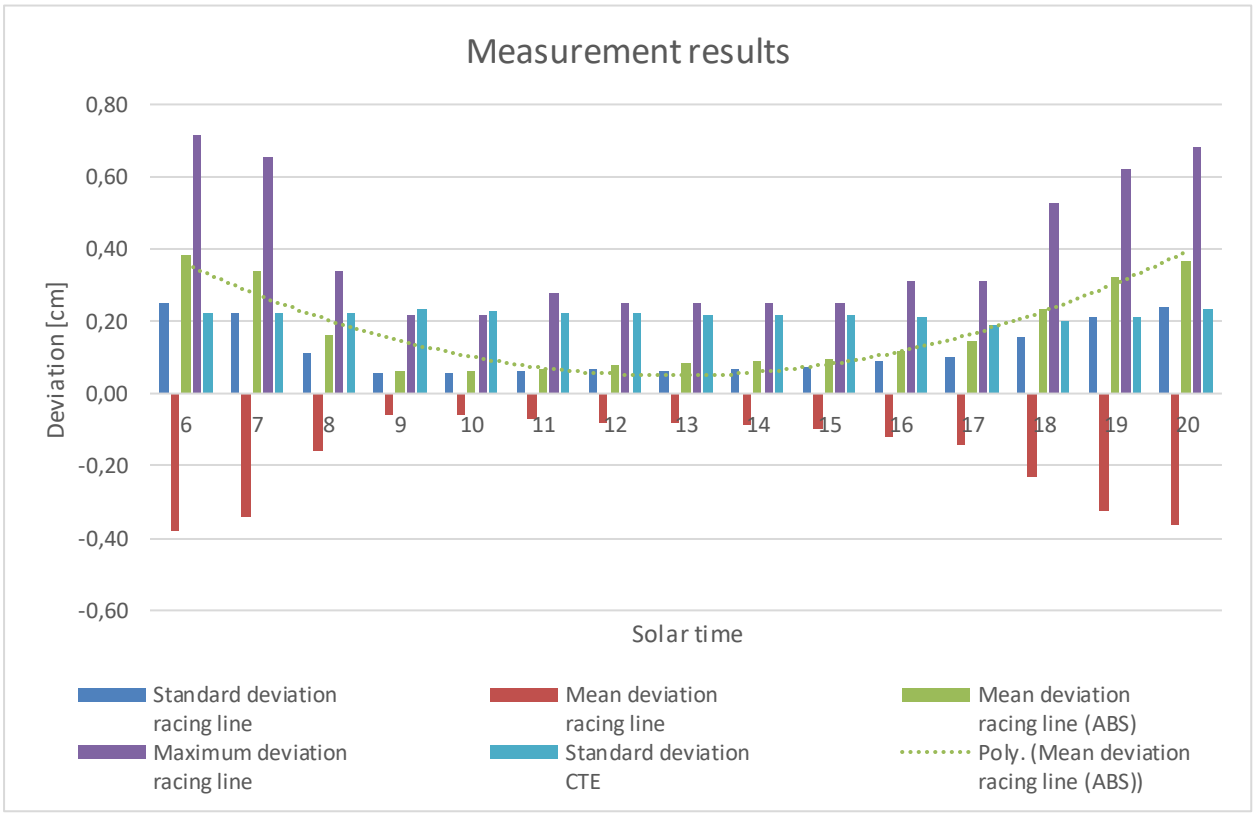
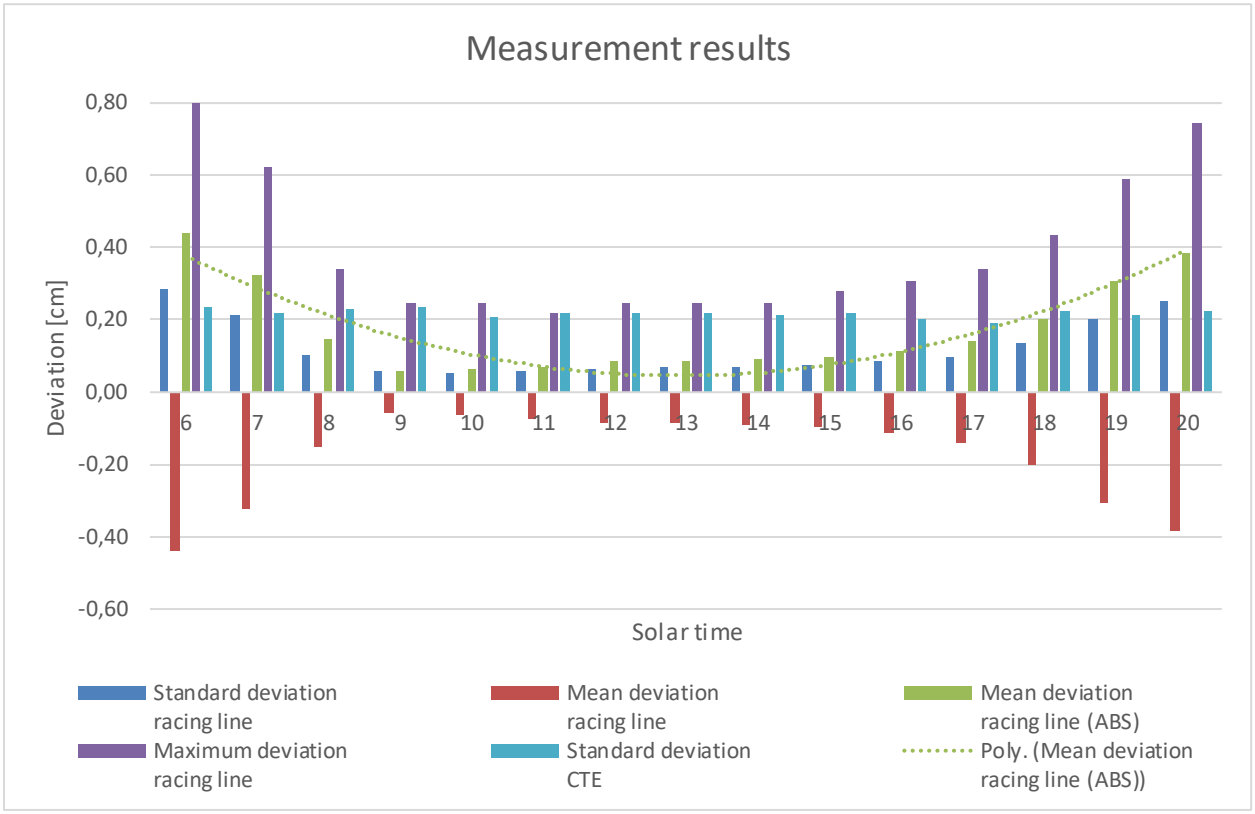


Abbildung 7-22: Messergebnisse simuliertes Datum, oben 1. Mai unten 15. Mai



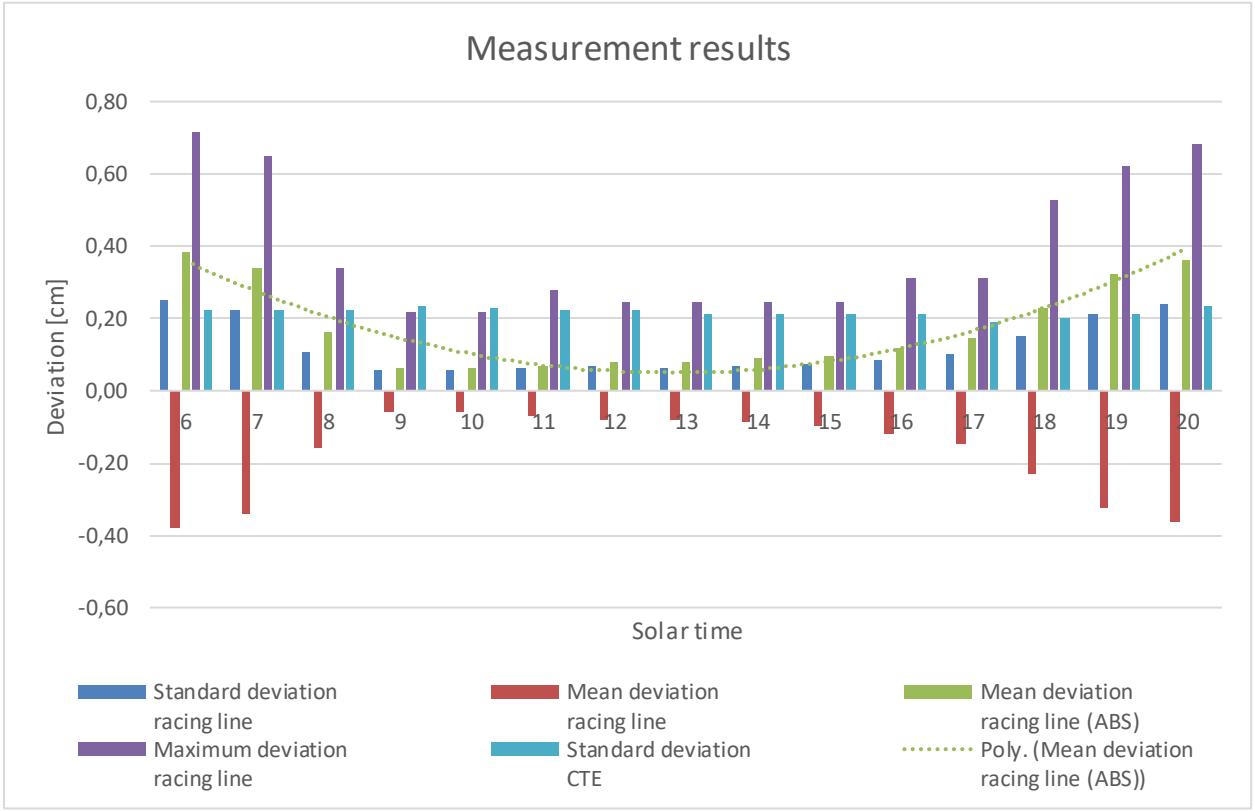


Abbildung 7-23: Messergebnisse simuliertes Datum, oben 1. Juni unten 15. Juni

7.3.5 Messreihe Penumbra

7.3.5.1 Beschreibung

Die Penumbra oder auch Halbschatten entsteht, wenn ein Objekt eine Lichtquelle teilweise verdeckt [62]. Das Objekt ist in unserem Fall das Zugfahrzeug, das in der „Unreal Engine“ nachgestellt wurde (siehe **Abbildung 7-24**).

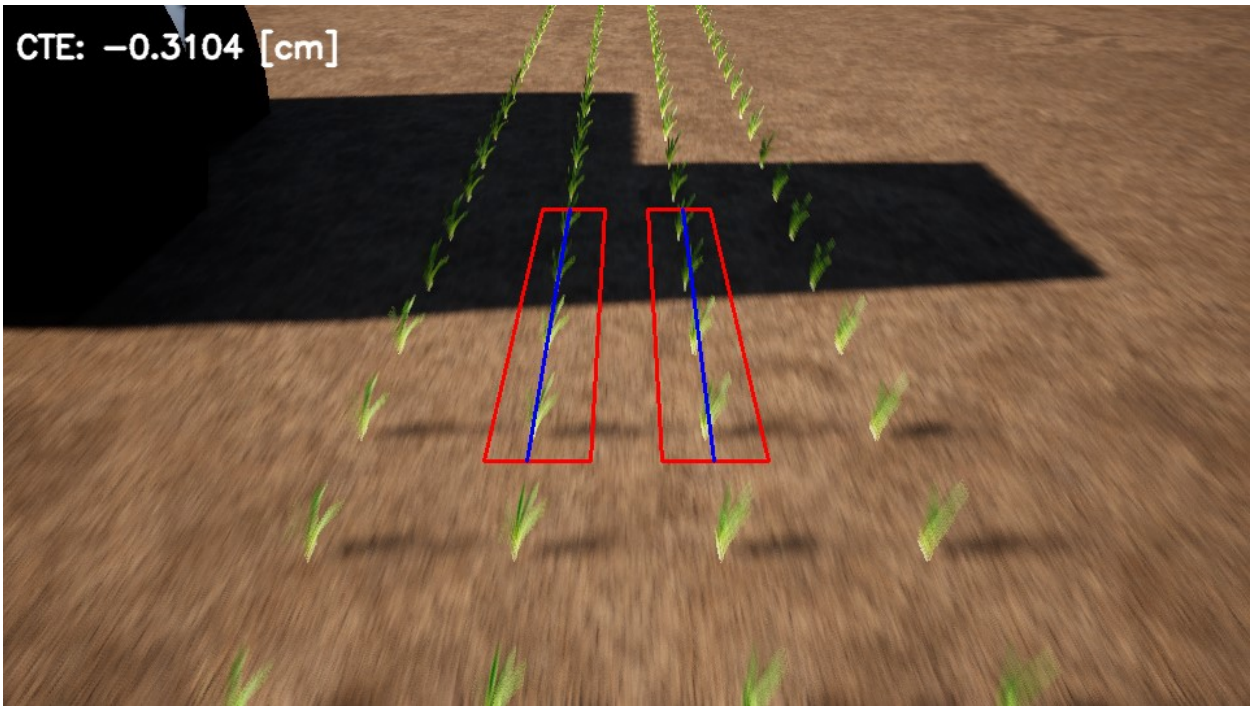


Abbildung 7-24: Virtuelle Umgebung mit Penumbra, Solar time: 8 Uhr

Um die Auswirkungen dieses Halbschattens zu untersuchen, erfolgten Tests zu unterschiedlichen Tageszeiten. Da die Abmessungen des Implements bisher unbekannt waren, wurden sie so festgelegt, dass die Pflanzenreihen im Lauf der Messreihe unter anderem im Halbschatten liegen. Die Pflanzen sind bei dieser Messreihe ohne Offset positioniert, um den Einfluss der Penumbra genauer analysieren zu können (vgl. Abschnitt 7.3.4). **Abbildung 7-25** und **Abbildung 7-26** zeigen das Zugfahrzeug in 3D-Darstellung und als Drahtgitter-Modell.

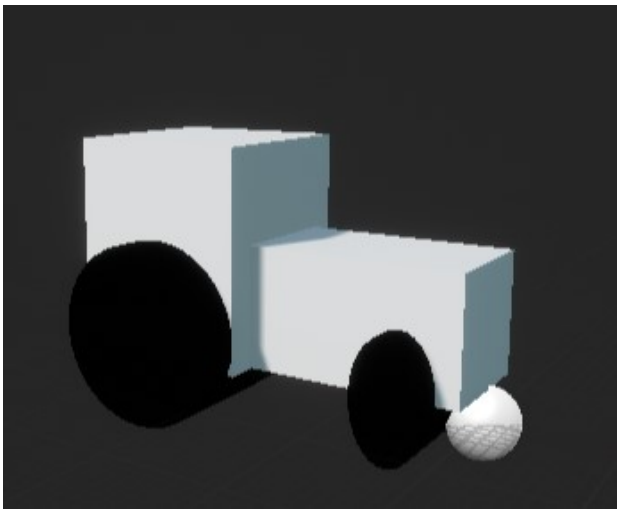


Abbildung 7-25: Zugfahrzeug 3D-Darstellung

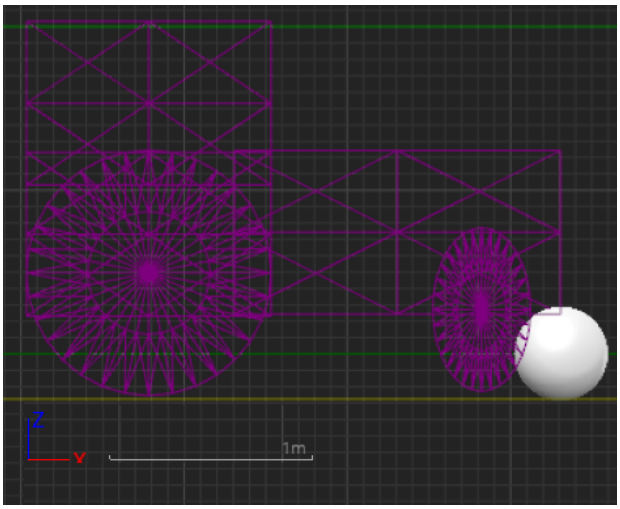


Abbildung 7-26: Zugfahrzeug Drahtgitter-Modell

7.3.5.2 Ergebnisse

In **Abbildung 7-27** sind die Messergebnisse der Simulation mit dem Halbschatten dargestellt. Da der Sonnenstand die Penumbra maßgeblich beeinflusst, ist es nicht verwunderlich, dass die Messergebnisse jenen aus dem Abschnitt Messreihen Sonnenstände ähneln. Bis ungefähr 9 Uhr liegen drei der vier Pflanzenreihen im Halbschatten des Zugfahrzeugs, was eine deutliche Verschlechterung des Messergebnisses im Vergleich zum vorangegangenen Abschnitt zur Folge hat. Ab 10 Uhr befindet sich nur mehr die linke Pflanzenreihe darin, was in den Ergebnissen deutlich zu sehen ist. Die besten Messergebnisse gibt es daher zwischen 10 und 17 Uhr. Danach werden die Aufnahmen deutlich weniger belichtet, was auch, wie schon in Abschnitt 7.3.4, wieder zu einem Anstieg der Abweichungen führt.

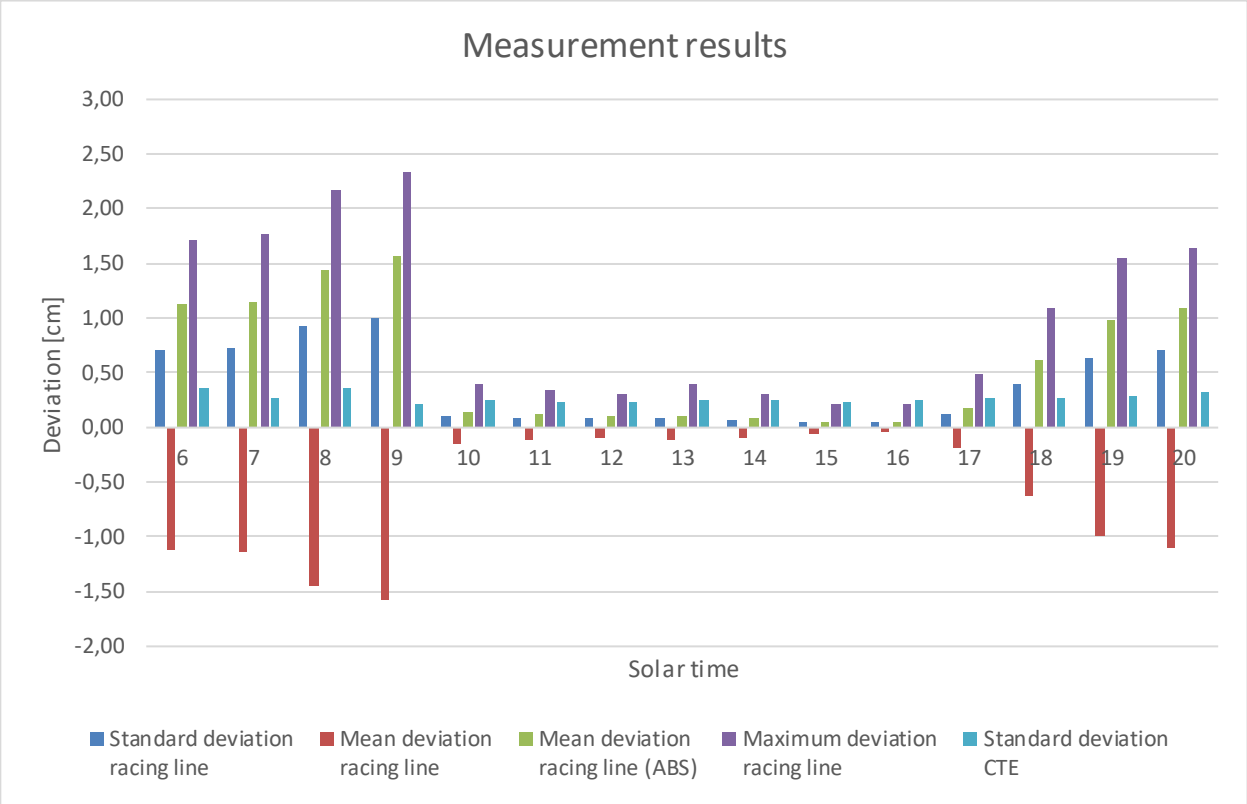


Abbildung 7-27: Messergebnisse Penumbra 1. Juni

7.3.6 Analyse der Farbabweichungen

Wie in den obigen Abschnitten (7.3.1-7.3.5) gezeigt wurde, beeinflussen Farbabweichungen die Genauigkeit des Algorithmus zwar spürbar, doch konnte das Konzept aus „Region of Interest", „Excess green index" und Otsu Schwellwertmethode in den meisten Simulationen überzeugen. In Abschnitt 7.3.1 wurde gezeigt, dass die Variation der Farbtemperatur kaum Einfluss auf das Messergebnis hat. Abschnitt 7.3.2 beschäftigte sich mit der Farbsättigung der Pflanzen und konnte zeigen, dass der Algorithmus zwar empfindlich auf eine zu geringe Farbsättigung reagiert, diese jedoch bei gesunden Pflanzen nicht vorkommt. In Abschnitt 7.3.3 konnte dann doch noch eine Schwachstelle aufgedeckt werden. Wird der Algorithmus mit Bildern stimuliert, die neben den Nutzpflanzen zusätzliche Grasflächen aufweisen, kommt es ab einer Grasfläche von rund 20% zu inakzeptablen Abweichungen. Die wohl spannendsten Messergebnisse liefern die nahezu identen Abschnitte 7.3.4 und 7.3.5, weshalb auf diese genauer einzugehen ist. Zu Beginn von Abschnitt 7.3.4.2 wurde erwähnt, dass die Charakteristik der Simulation des 1. Mai nicht zu den restlichen Simulationen passt. Der Grund dafür war, dass die Messergebnisse gegen Abend (19 und 20 Uhr) wieder besser wurden. Um der Ursache auf den Grund zu gehen, werden in **Abbildung 7-28** die „Regions of Interest" nach Schritt 4 des Algorithmus (vgl. **Abbildung 4-2**), der Otsu Schwellwertmethode, dargestellt. Die mittlere absolute Abweichung zwischen der Messung um 18 Uhr und jener um 20 Uhr beträgt 18 mm, weshalb die Ursache dafür im Detail gesucht

werden muss. Ein möglicher Grund für diese Abweichung ist bereits in **Abbildung 7-28** dargestellt. Zwischen 18 und 19 Uhr wird das linke untere Blatt der untersten Pflanze, das nur mehr aus wenigen Pixeln besteht, anders dargestellt. Dieses Detail ist in **Abbildung 7-28** zuerst mittels eines farbigen Kreises hervorgehoben und anschließend vergrößert dargestellt. Eine derartige Veränderung kann bereits den Unterschied der Simulation vom 1. Mai zu den anderen verursachen.

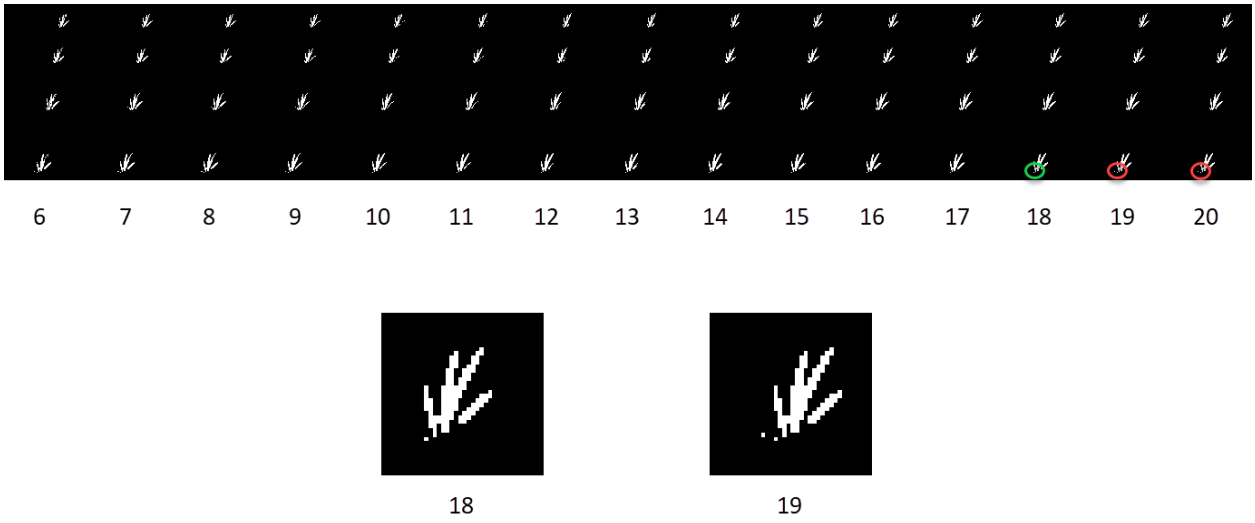


Abbildung 7-28: ROI nach Otsu Schwellwertmethode 1. Mai von 6 Uhr bis 20 Uhr

Abbildung 7-29 wurde erstellt, um die Charakteristik der Messergebnisse aus Abschnitt 7.3.4 zu verstehen. Sie zeigt jeweils dieselbe Pflanze, dargestellt zu unterschiedlichen Uhrzeiten. Der markanteste Unterschied in den einzelnen Darstellungen ist das rechte Blatt der Pflanze, das ab 10 Uhr langsam verschwindet. Ab 13 Uhr tritt es allmählich wieder ins Bild. Eine derartige Veränderung hat bei durchschnittlich 57 Pixeln pro Darstellung bereits deutliche Auswirkungen auf die Messergebnisse, wie **Abbildung 7-23** und **Abbildung 7-28** zeigen.

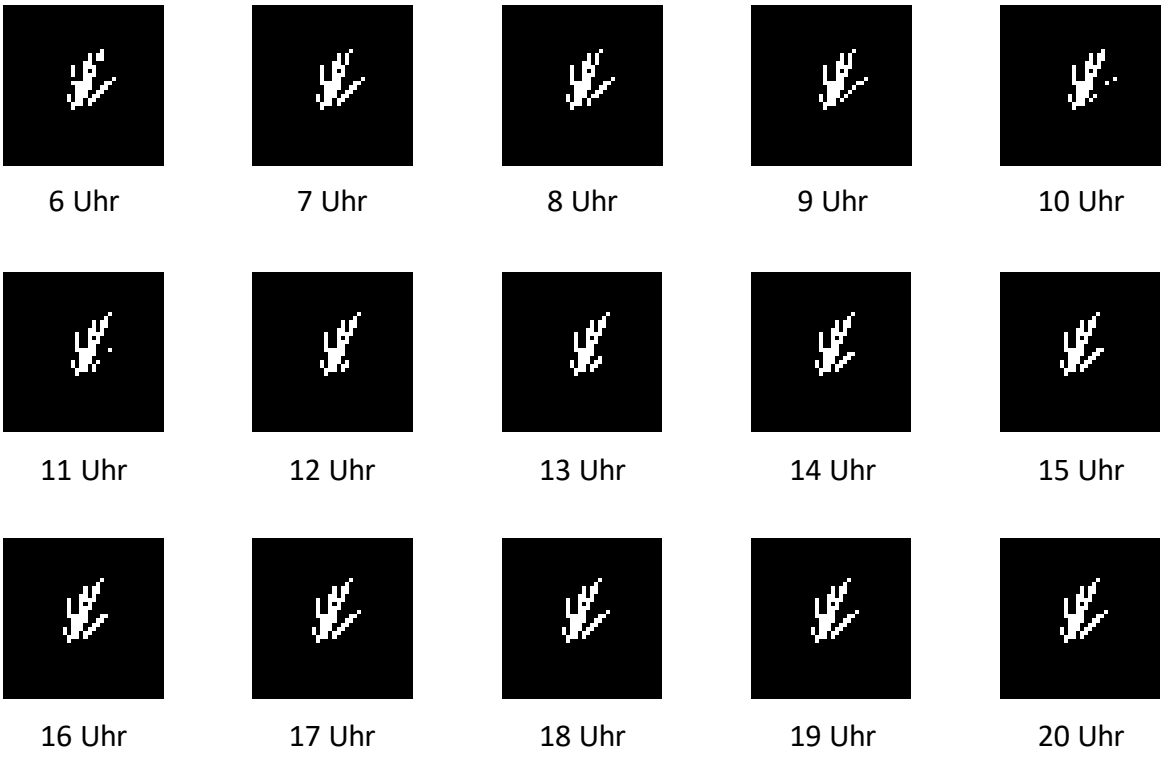


Abbildung 7-29: Binärbilder 1. Mai von 6 bis 20 Uhr

Die **Abbildung 7-30** wird als Gleichung angedeutet und dient dazu, das eben Erwähnte graphisch nochmals darzustellen. Sie zeigt dieselbe Pflanze einmal um 7 Uhr und einmal um 12 Uhr und soll die Unterschiede der beiden Darstellungen aufzeigen.

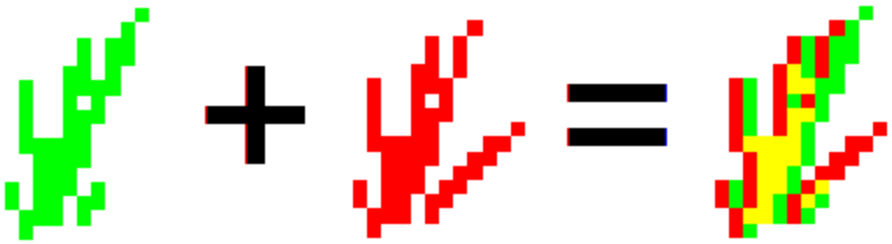


Abbildung 7-30: Bitmap derselben Pflanze rot um 7 Uhr und grün um 12 Uhr

7.4 Robustheit gegenüber geometrischen Einflüssen

Diese Messreihen sollen die Robustheit des Algorithmus gegenüber geometrischen Einflüssen untersuchen. Dazu werden neben den extrinsischen Parametern der Kamera auch die Pflanzenreihen variiert.

7.4.1 Messreihe mit Kameraerschütterungen

7.4.1.1 Beschreibung

Ist ein Traktor auf einem Feld unterwegs, kommt es unweigerlich zu Erschütterungen. Moderne Zugfahrzeuge sind heutzutage mit Luftsitzen ausgestattet, um diese Stöße zu dämpfen und damit den Komfort für den/die FahrerIn zu erhöhen. Ähnliche Vorrichtungen für die Montage von Kamerasystemen sind bis heute in der Landwirtschaft nicht verbreitet, weshalb die Erschütterungen direkt auf die Kamera übertragen werden. Zwar besitzen viele Kameras Bildstabilisatoren, doch zeigen Vergleiche mit realen Bildern, dass die Erschütterungen in den Aufnahmen noch deutlich zu erkennen sind. Dieser Abschnitt untersucht daher die Auswirkungen derartiger Kameraerschütterungen, indem zur aktuellen Kameraposition ein Vektor addiert wird. Die Verstärkung kann über die Variable „Gain_Shake“ in der „Unreal Engine“ eingestellt werden. Ein Gain von 0 entspricht keiner Erschütterung. Bei einem Gain von 1 wird beispielsweise zur X-Position eine Zufallszahl zwischen ± 1 cm addiert. Dasselbe geschieht auch mit der Y- und Z-Position. Die Frequenz der Kameraerschütterung ist ident mit der Bildwiederholungsrate, die für diese Simulation mit 15 FPS festgelegt wurde.

7.4.1.2 Ergebnisse

Die Ergebnisse der aktuellen Messreihe zeigt **Abbildung 7-31**. Wie zu erwarten, nehmen alle statistischen Kenngrößen bis auf die relative Mittlere Abweichung mit zunehmender Kameraerschütterung zu. Der Grund für die verhältnismäßig geringe Zunahme der Abweichungen liegt einerseits darin, dass das träge Implement auf die hohe Frequenz der Erschütterungen nicht reagieren kann, und andererseits darin, dass der addierte Offset im Mittel Null ergibt.

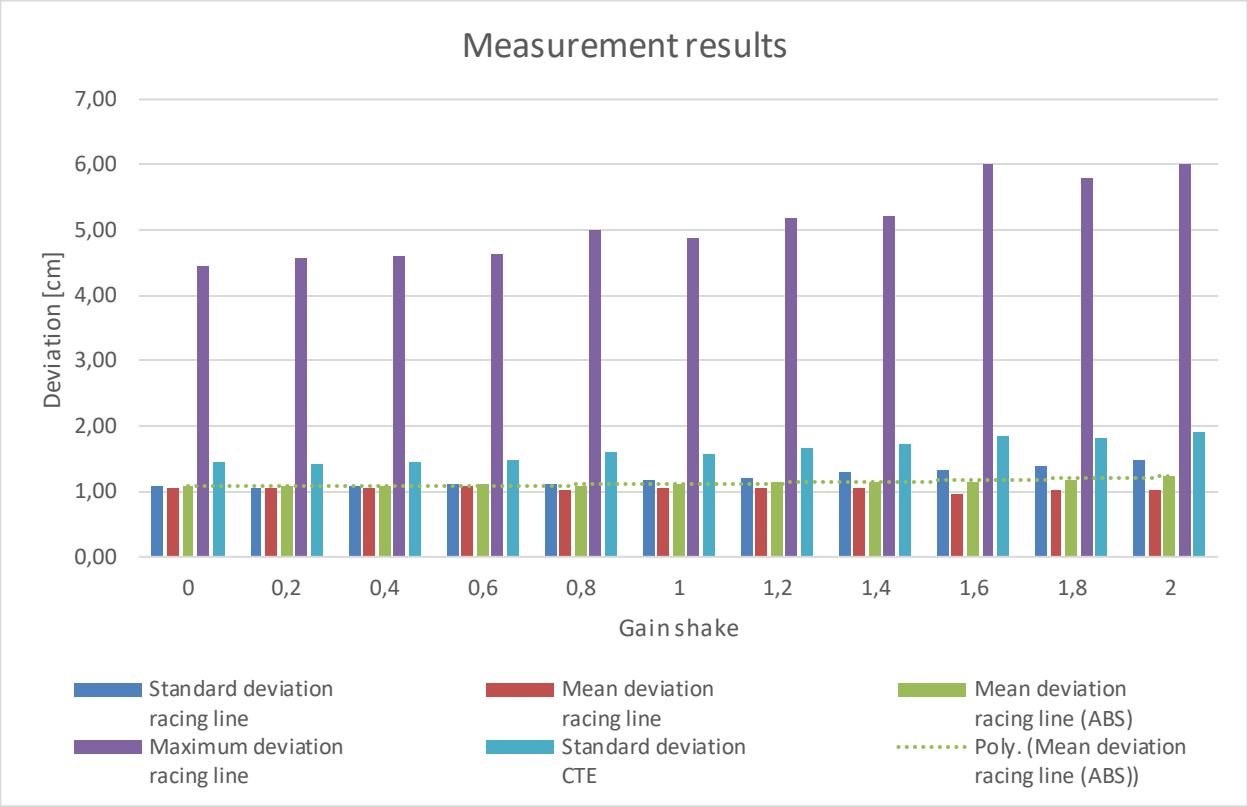


Abbildung 7-31: Messergebnisse Kameraerschütterungen

7.4.2 Messreihen Skalierung der Pflanzen

7.4.2.1 Beschreibung

Mithilfe der Skalierung sollen in diesen Messreihen unterschiedlichste Wachstumsstadien der Pflanzen simuliert werden. Eine Skalierung der Pflanzen spiegelt in der Realität nur in den seltensten Fällen ein einigermaßen reales Wachstum der Pflanzen wider (vgl. 5.3.4.1). Dennoch sollen in den folgenden zwei Messreihen die Skalierungen der Pflanzen schrittweise von 10% auf 100% erhöht werden, um die Auswirkungen auf den Algorithmus zu evaluieren. Um störende Einflüsse zu vermeiden, ist der Messaufbau so simpel wie möglich gehalten. Die Pflanzen werden nicht zufällig rotiert und auch der Offset zwischen den Pflanzen ist auf Null gesetzt. Die beiden folgenden Messreihen unterscheiden sich lediglich in der Rotation der Pflanzen. Die erste wurde ohne Rotation durchgeführt (vgl. **Abbildung 7-32**), bei der zweiten wurden die Pflanzen um 180° gedreht (vgl. **Abbildung 7-33**).

Abbildung 7-32: Rotation 0°

Abbildung 7-33: Rotation 180°

7.4.2.2 Ergebnisse

Abbildung 7-34 zeigt, dass die Skalierung der Pflanzen einen maßgeblichen Einfluss auf die Genauigkeit des Algorithmus hat. Dass die Beträge der relativen und absoluten mittleren Abweichung ident sind, aber ein anderes Vorzeichen haben, sagt aus, dass die gefahrene Linie über die gesamte Simulationsdauer hinweg rechts von der Ideallinie liegt. Diese Abweichung ist auf den rechts aus der Mitte verschobenen Schwerpunkt der Pflanze zurückzuführen (vgl. Abschnitt 7.4.5).

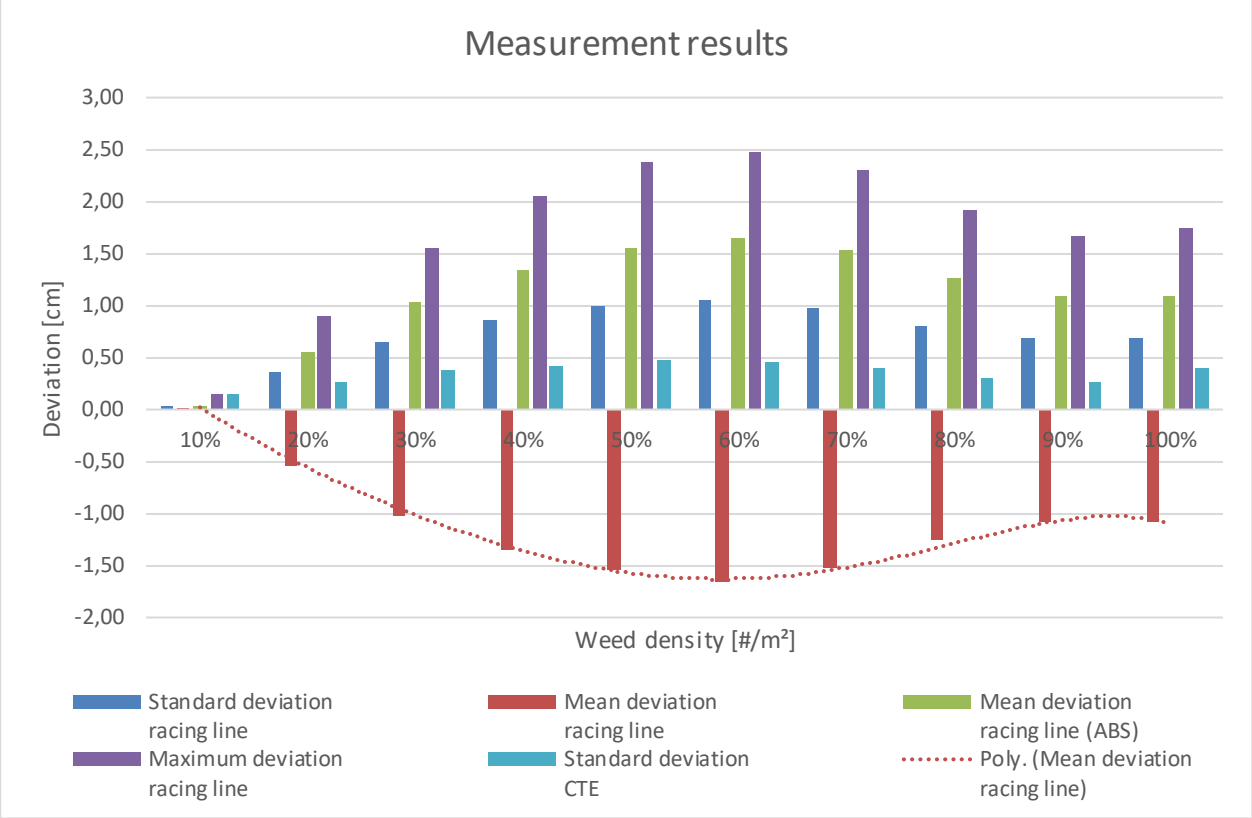


Abbildung 7-34: Messergebnisse Skalierung der Pflanzen, Rotation 0°

Abbildung 7-35 bildet die Messergebnisse der zweiten Messreihe ab, bei der die Pflanzen um 180° gedreht wurden. Im Gegensatz zur **Abbildung 7-34** haben sowohl die relative als auch die absolute mittlere Abweichung ein positives Vorzeichen. Umgekehrt wie in der ersten Messreihe, lässt sich daraus schließen, dass sich die gefahrene Linie stehts links von der Ideallinie befindet.

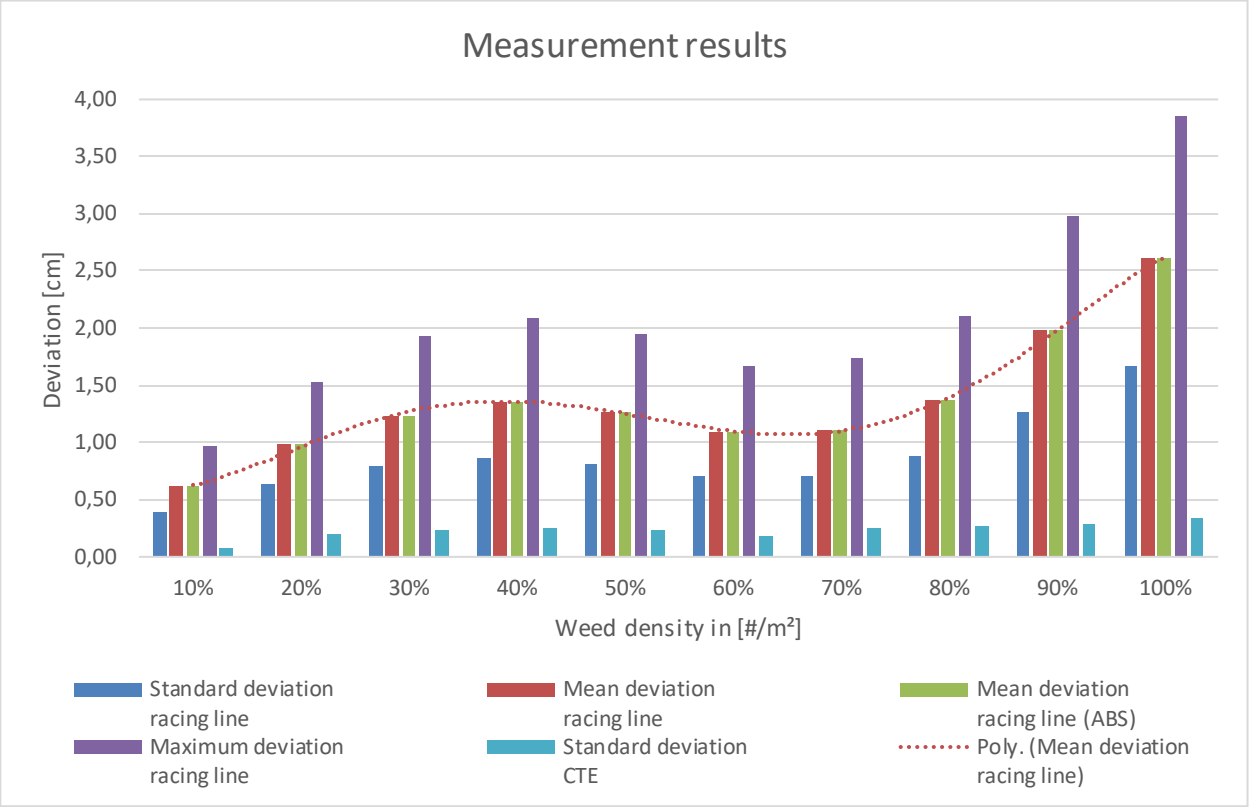


Abbildung 7-35: Messergebnisse Skalierung der Pflanzen, Rotation 180°

7.4.3 Messreihen mit Rotationen der Kamera

7.4.3.1 Beschreibung

Aktuell werden die extrinsischen Parameter der Kamera vor der Messung auf dem Feld festgelegt und anschließend nicht mehr verändert. Die folgende Messreihe soll Aufschluss darüber geben, wie sich eine Abweichung im Roll-, Nick- und Gierwinkel auf das Messergebnis auswirkt.

7.4.3.2 Ergebnisse

Die Messergebnisse aus **Abbildung 7-36** entsprechen den Erwartungen. Die Abweichungen zwischen Ideallinie und tatsächlich gefahrener Linie sind bei einem Gierwinkel von 0° Grad am geringsten und nehmen mit zu- beziehungsweise abnehmendem Winkel zu. Für jene Messung mit einem Gierwinkel von + 5 ° sind die Messergebnisse nicht graphisch dargestellt, da der Algorithmus nicht mehr in der Lage ist, die Simulation ordnungsgemäß zu Ende zu führen.

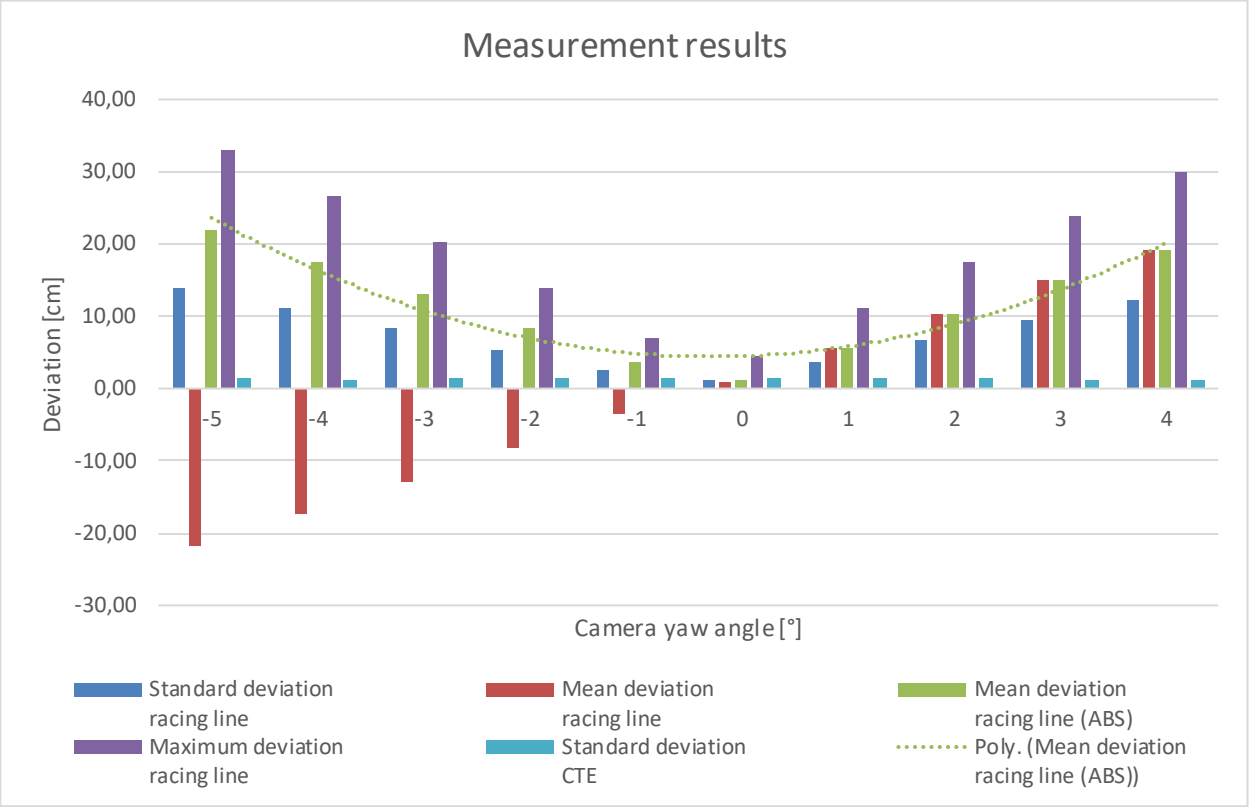


Abbildung 7-36: Messergebnisse Variation Gierwinkel

Abbildung 7-37 zeigt jene Messergebnisse, bei denen der Nickwinkel im Lauf der Simulationen verändert wird. Das Auffälligste an diesen Ergebnissen ist, dass die Standardabweichung des „Cross Track Errors“ mit zunehmend falscherem Winkel abnimmt. Dieses Phänomen ist auf die perspektivische Darstellung, wie sie bei Aufnahmen mit Kameras vorkommt, zurückzuführen. Ein und dieselbe Positionsveränderung wirkt in der Ferne nicht so erheblich, wie wenn sie unmittelbar vor dem Betrachter vonstattengeht. Im vorliegenden Fall wird die Position der „Region of Interest“, die sich mit zunehmend flacherem Winkel immer näher an der Kamera befindet, verändert. Da der CTE in einem unmittelbaren Zusammenhang mit dieser Positionsänderung steht (vgl. [9, Ch. 4.6]), erklärt das den abnehmenden Verlauf.

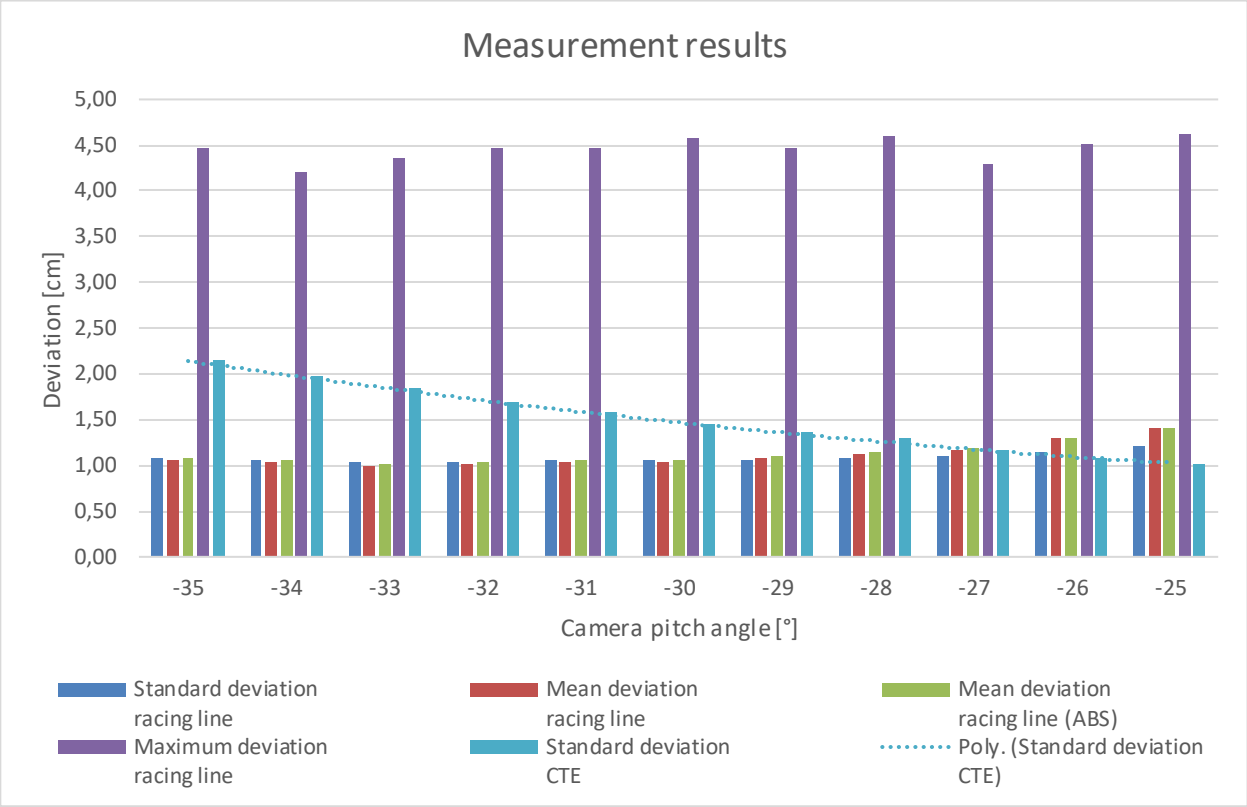
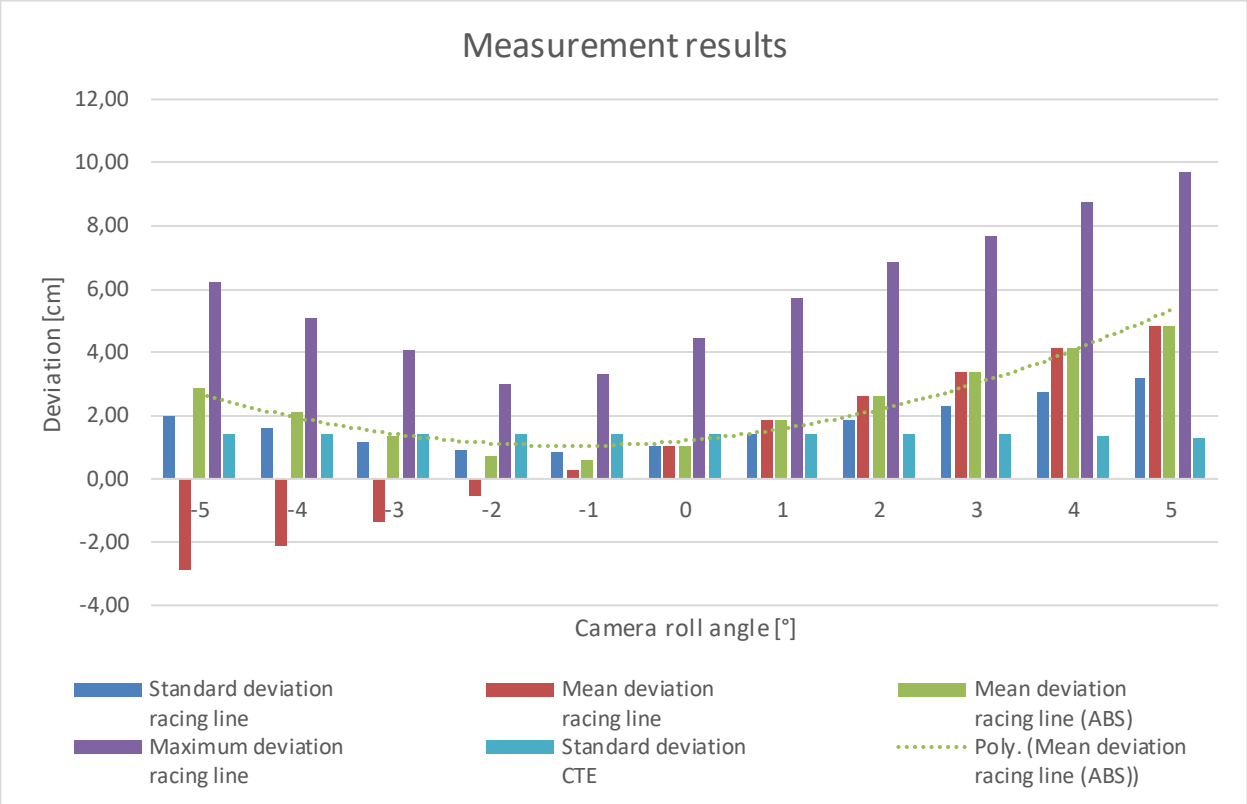


Abbildung 7-37: Messergebnisse Variation Nickwinkel

Die Messergebnisse aus **Abbildung 7-38** (oben) entsprechen nicht jenen, die man von einer Variation des Rollwinkels erwartet. Aus diesem Grund wird die Messreihe mit 15 ± 2 cm Pflanzenabstand in **Abbildung 7-38** (unten) wiederholt. Die Resultate entsprechen nun den vermuteten Ergebnissen. Ähnlich wie bei der Variation des Gierwinkels (vgl. **Abbildung 7-36**), ist die Abweichung bei 0° am kleinsten. Mit zu- beziehungsweise abnehmendem Rollwinkel nimmt die Abweichung linear zu. Zwar ähnelt die Charakteristik dieser Messreihe jener aus **Abbildung 7-36**, doch ist die absolute mittlere Abweichung annähernd um den Faktor 5 kleiner.



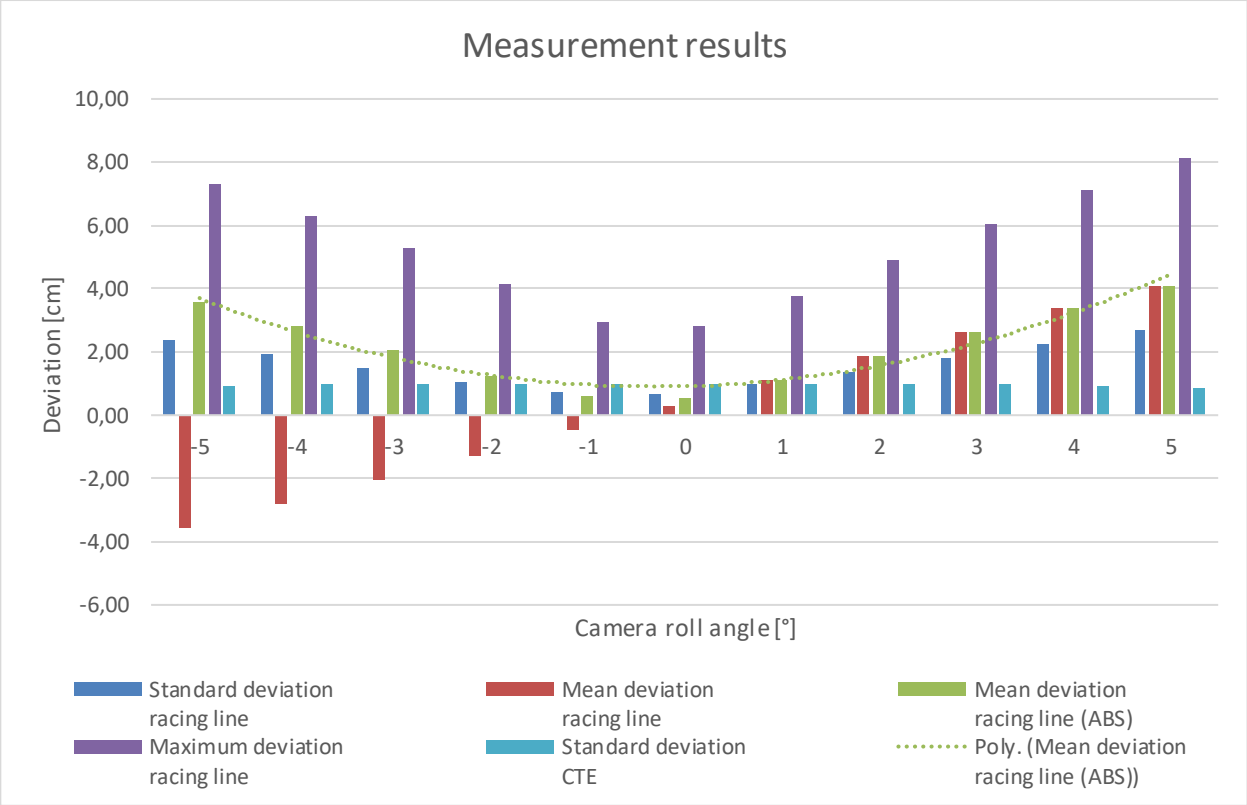


Abbildung 7-38: Messergebnisse Variation Rollwinkel (50 cm Pflanzenabstand oben, 15 cm unten)

7.4.4 Messreihe Kurvenfahrt

7.4.4.1 Beschreibung

Nicht immer ist die Pflanzenreihe eine gerade Linie, weshalb der Algorithmus auch in Kurven die Spur halten können sollte. Für diese Messreihe sind die Pflanzen in der „Unreal Engine“ so positioniert, dass sich zwei Halbkreise (einer mit einem Radius von 25 m und der andere mit einem Radius von 25,5 m) ergeben. Auf den Offset der Pflanzen wird vorerst verzichtet, da nicht davon auszugehen ist, dass der Algorithmus mit seiner aktuellen Konfigurierung in der Lage ist, diesen Halbkreis zu durchfahren.

7.4.4.2 Ergebnisse

Wie vermutet, ist der Algorithmus aktuell noch nicht fähig, eine Kurve fehlerfrei zu durchfahren. Wie sich in **Abbildung 7-39** erkennen lässt, liegt die gefahrene Linie deutlich zu weit innen. Das schlechte Abschneiden des Algorithmus in einer Kurve liegt einerseits an der relativ langen „Region of Interest“ (ROI) und andererseits am flachen Kamerawinkel von -30°. Eine Veränderung dieser Parameter würde zu besseren Ergebnissen führen, doch würden diese sich auch negativ bei anderen Messreihen, wie zum Beispiel jener aus Abschnitt 7.1.3 auswirken.

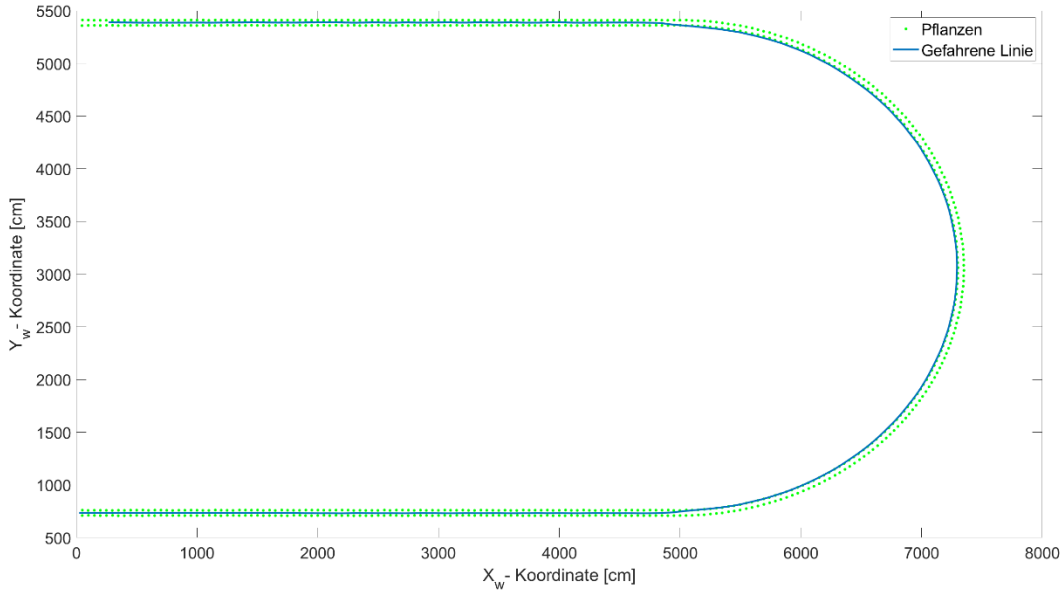


Abbildung 7-39: Simulation Kurvenfahrt

7.4.5 Analyse der geometrischen Einflüsse

In den Abschnitten 7.4.1 bis 7.4.4 konnte gezeigt werden, dass geometrische Einflüsse einen wesentlichen Einfluss auf die Genauigkeit des Algorithmus haben. Begonnen wurde in Abschnitt 7.4.1 mit der Kameraerschütterung, welche im Vergleich zu den restlichen Messreihen eine verhältnismäßig geringe Auswirkung auf die Messergebnisse hat. In Abschnitt 7.4.2 wurde versucht, das Wachstumsstadion der Pflanzen mittels Skalierung zu simulieren. Dabei kam es bereits zu größeren Abweichungen in den Messergebnissen, welche nachfolgend analysiert werden sollen.

Wie bereits angedeutet, ist der Grund für die schwankenden Messergebnisse aus Abschnitt 7.4.2 darauf zurückzuführen, dass der vom Algorithmus berechnete Schwerpunkt der Pflanze nicht mit deren Ursprung übereinstimmt. Für die Darstellung der Pflanze in **Abbildung 7-40** wurde eine Aufnahme in der virtuellen Umgebung gemacht, die Pflanze mittels ExG und Otsu Schwellwertmethode vom Boden separiert und mithilfe von „Matlab“ der Schwerpunkt der Pflanze berechnet. Wie sofort erkennbar ist, stimmen Schwerpunkt und Ursprung nicht überein.

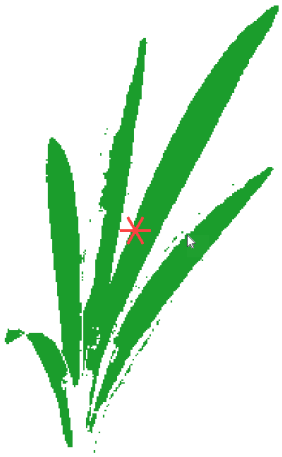


Abbildung 7-40: Berechneter Schwerpunkt einer Pflanze

Da die Pflanze in X- Y- und Z-Richtung gleichermaßen skaliert wird, sollte sich der Schwerpunkt, unabhängig von der Skalierung, relativ zum Ursprung stets an derselben Position der Pflanze befinden. Absolut betrachtet wird der Abstand zwischen Ursprung und Schwerpunkt mit zunehmender Skalierung größer, was die Zunahme der Messabweichung in **Abbildung 7-32** und **Abbildung 7-33** erklärt. Nicht geklärt ist jedoch bislang das Auf- und Abschwingen der Messergebnisse. **Abbildung 7-41** zeigt dazu fünf „Regions of Interest“ mit unterschiedlich skalierten Pflanzen. Mithilfe der „Linearen Regression“ wird versucht, die Pflanzenreihe zu detektieren. Wie man gut erkennen kann, ändert sich die detektierte Linie mit zunehmender Skalierung. Ab einer gewissen Skalierung kommt es vor, dass sich die Pflanzen aufgrund der perspektivischen Geometrie optisch überlagern. Die Bestimmung des Schwerpunkts für eine

konkrete Pflanze ist dann nicht mehr möglich was die Messergebnisse aus **Abbildung 7-34** und **Abbildung 7-35** erklärt.

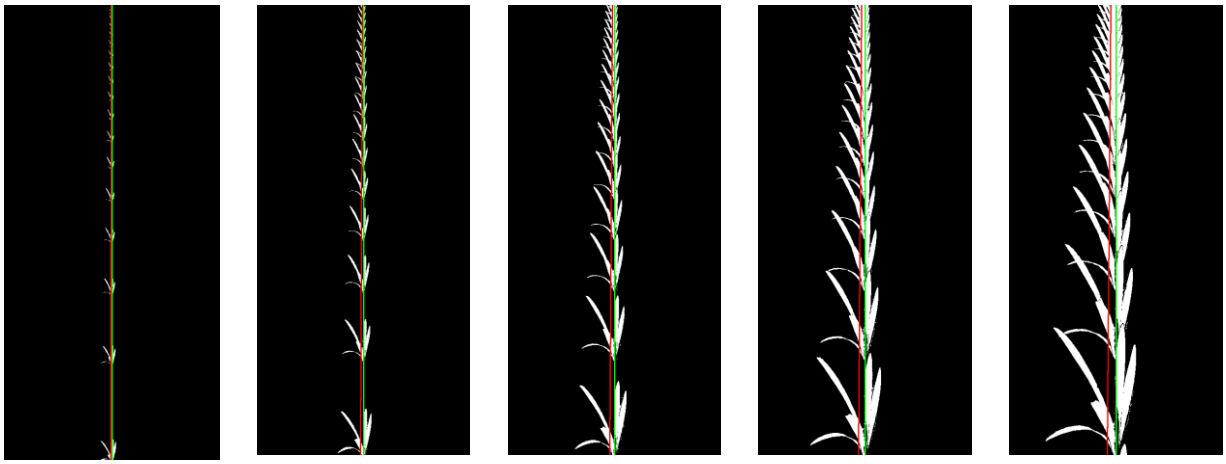


Abbildung 7-41: Binärbilder mit unterschiedlichen Skalierungen der Pflanzen (rot detektierte Linie grün Ideallinie)

Abschnitt 7.4.3 liefert bislang die schlechtesten Ergebnisse dieser Arbeit. Mit maximalen Abweichungen von über 30 cm, bei der Rotation des Gierwinkels, sind diese Messergebnisse wohl für die wenigsten Anwendungen akzeptabel. Vermutet wird der Grund dieser großen Abweichungen in der fehlenden Rückführung der Eulerwinkel in den Algorithmus. Bislang wird der Roll-, Nick- und Gier-Winkel zu Beginn der Simulation sowohl in der „Unreal Engine“ als auch im Algorithmus festgelegt und nicht mehr verändert. Die **Abbildung 7-42** und **Abbildung 7-43** zeigen ein und dieselbe Szene einer ursprünglich mittig durch das Bild verlaufenden Pflanzenreihe. Dieser Aufnahme, welche mit 0° Gier- und Roll-Winkel aufgenommen wurde, werden in **Abbildung 7-42** zwei weitere Aufnahmen mit einem Gierwinkel von -5° beziehungsweise +5° überlagert. Dasselbe wurde in **Abbildung 7-43** für den Rollwinkel durchgeführt. Wie deutlich zu erkennen ist, wird die Position der Pflanzen vollkommen falsch eingeschätzt, was die fatalen Ergebnisse aus Abschnitt 7.4.3 erklärt. In Abschnitt 7.4.4 wurde abschließend noch eine Kurvenfahrt simuliert. Die Ergebnisse konnten, wie bereits erwähnt, aufgrund der zu langen ROI und des flachen Kamerawinkels nicht überzeugen.

Allgemein konnte in Abschnitt 7.4 gezeigt werden, dass geometrische Einflüsse drastische Auswirkungen auf die Messergebnisse haben, doch lassen sich gerade in der Simulation derartige Szenarien bestens nachstellen.

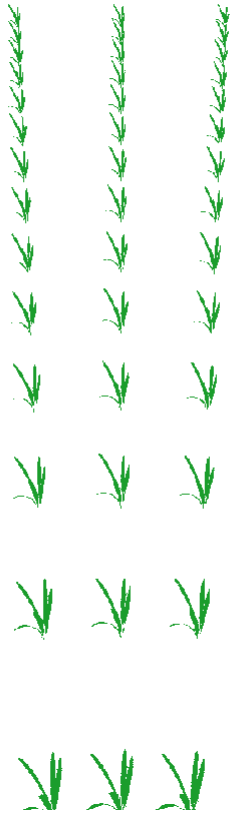


Abbildung 7-42: Variation Gierwinkel

Abbildung 7-43: Variation Rollwinkel

7.5 Robustheit gegenüber Unkraut

Der vorliegende Algorithmus wurde entwickelt, um den/die LandwirtIn bei der mechanischen Unkrautbekämpfung zu unterstützen. Ob der Algorithmus imstande ist, diese Aufgabe ordnungsgemäß durchzuführen, soll die folgende Messreihe klären. Dieser Abschnitt zählt zu jenen, bei denen zusätzlich zu 50 ± 2 cm auch mit 15 ± 2 cm Pflanzenabstand simuliert wird, da eine deutliche Veränderung im Messergebnis anzunehmen ist.

7.5.1 Messreihe ohne „Blob reduction“

7.5.1.1 Beschreibung

In der „Unreal Engine“ wurde für diese Messreihe ein „Blueprint“ mit dem Namen „Weed_Placement“ entwickelt (vgl. 5.3.1). Dieses „Blueprint“ erlaubt es, die Unkrautdichte beliebig festzulegen. Für diese Messreihe wird die Unkrautdichte von 0 auf 14 Stück pro Quadratmeter schrittweise erhöht. Das Inkrement beträgt 2 Stück pro Quadratmeter, was bedeutet, dass

insgesamt 8 Messungen stattfinden. **Abbildung 7-44** zeigt die virtuelle Umgebung mit 50 ± 2 cm Pflanzenabstand und starkem Unkrautbewuchs.

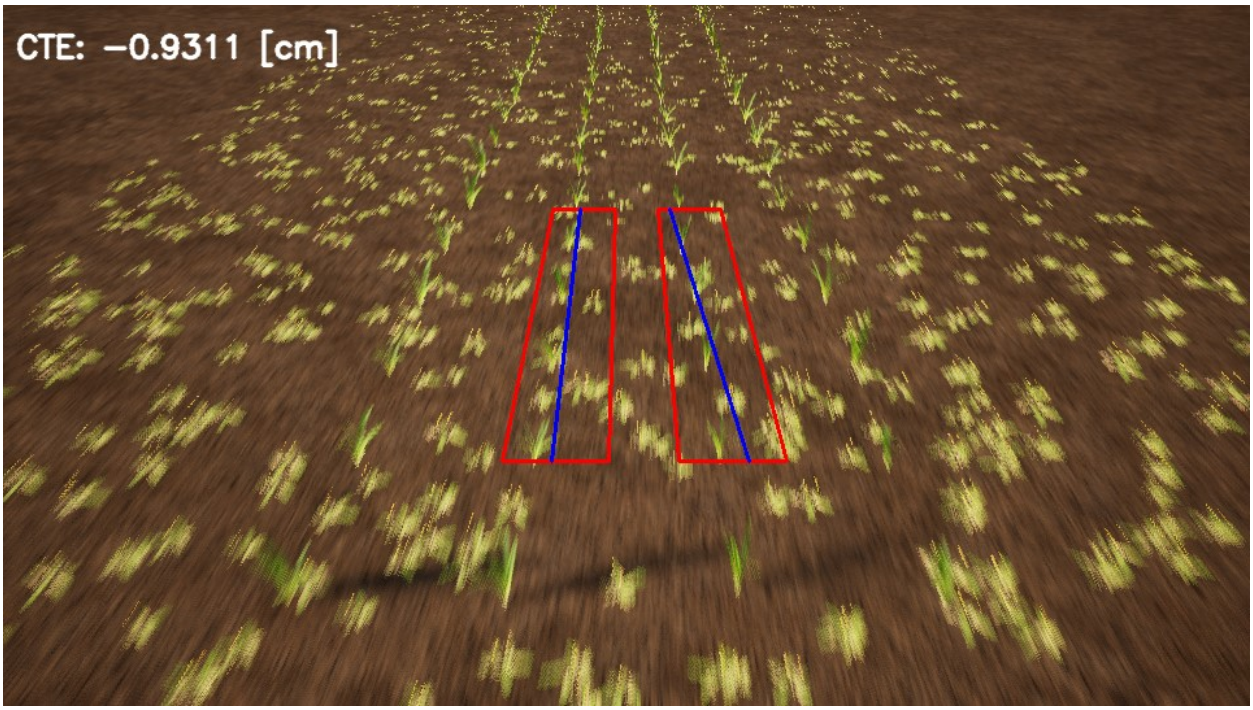


Abbildung 7-44: Virtuelle Umgebung mit einer Unkrautdichte von 14 Stück/m²

7.5.1.2 Ergebnisse

Bei einer Unkrautdichte von 0 bis 10 Stück pro Quadratmeter (Messung 1-6) kommt es zwar zu einem deutlichen Anstieg der Abweichungen, der Algorithmus ist aber dennoch imstande, die komplette Simulationsdauer hindurch die Spur zu halten. Aufgrund der Ähnlichkeit dieser Ergebnisse mit jenen aus **Abbildung 7-46** wird auf eine graphische Darstellung verzichtet. Ab einer Unkrautdichte von 12 Stück pro Quadratmeter kommt es bei den durchgeführten Simulationen bereits zum Spurwechsel. **Abbildung 7-45** zeigt das Resultat einer Simulation, die mit einer Unkrautdichte von 14 Stück pro Quadratmeter und einem Pflanzenabstand von 50 ± 2 cm erfolgte. Wie darauf zu erkennen ist, wechselt der Algorithmus nach etwa 90 m die Spur.

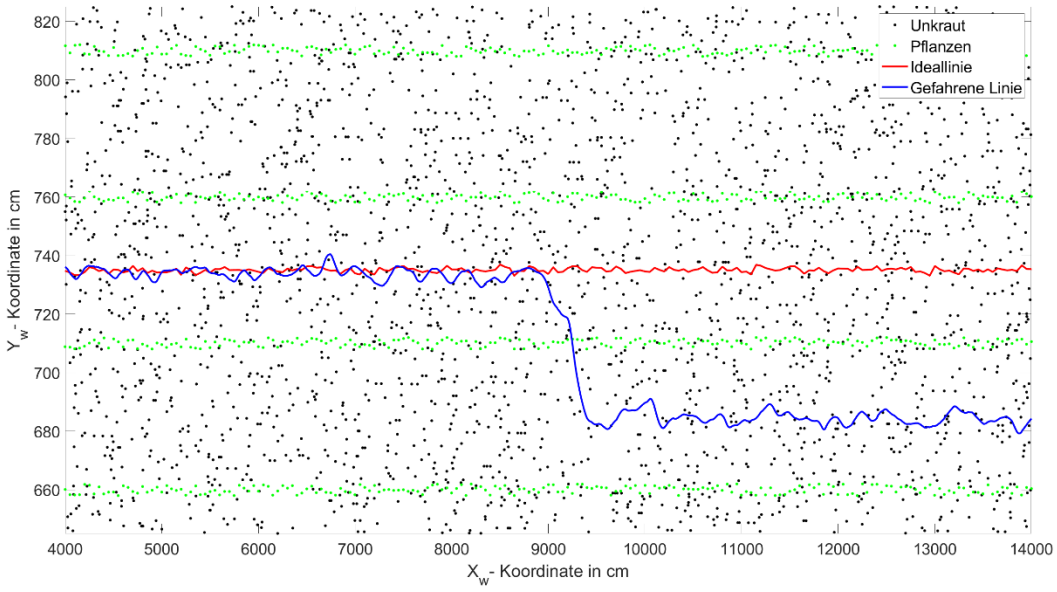


Abbildung 7-45: Simulation ohne „Blob reduction“ (50 ± 2 cm Pflanzenabstand)

7.5.2 Messreihe mit „Blob reduction“

7.5.2.1 Beschreibung

Um die Performance des Algorithmus weiter zu erhöhen, wurde eine „Blob reduction“ implementiert. Die Idee dahinter ist, dass Unkraut im binären Bild als kleine weiße „Blobs“ dargestellt ist. Im Gegensatz dazu werden Pflanzen als mehrere zusammenhängende weiße Pixel dargestellt. Im Algorithmus kann eine untere Schranke angegeben werden, die festlegt, ab wann ein Verbund an weißen zusammenhängenden Pixeln als Pflanze gewertet wird. Die anderen weißen Pixel werden nicht als Pflanze eingestuft und somit verworfen. Der Schwellwert muss an die Umgebungsbedingungen angepasst sein, um nicht versehentlich Pflanzen im frühen Wachstumsstadium als Unkraut zu interpretieren [9, Ch. 4.4]. In unserem Beispiel liegt dieser Schwellwert bei 50 Pixel.

7.5.2.2 Ergebnisse

Die Ergebnisse der Simulationen mit „Blob reduction“ finden sich in **Abbildung 7-46**. Wie daraus zu entnehmen ist, ist der Algorithmus mit „Blob reduction“ in der Lage, alle simulierten Szenarien positiv zu absolvieren. Ansatzweise vergleichbar mit der Messreihe ohne „Blob reduction“, steigen die Abweichungen mit zunehmender Unkrautdichte an. Eine Erhöhung der absoluten mittleren Abweichung um 75% zwischen erster und letzter Messung ist zwar relativ viel, doch liefert der Algorithmus auch bei sehr starkem Unkrautbewuchs mit einer absoluten mittleren Abweichung von 1,19 cm noch immer sehr gute Ergebnisse.

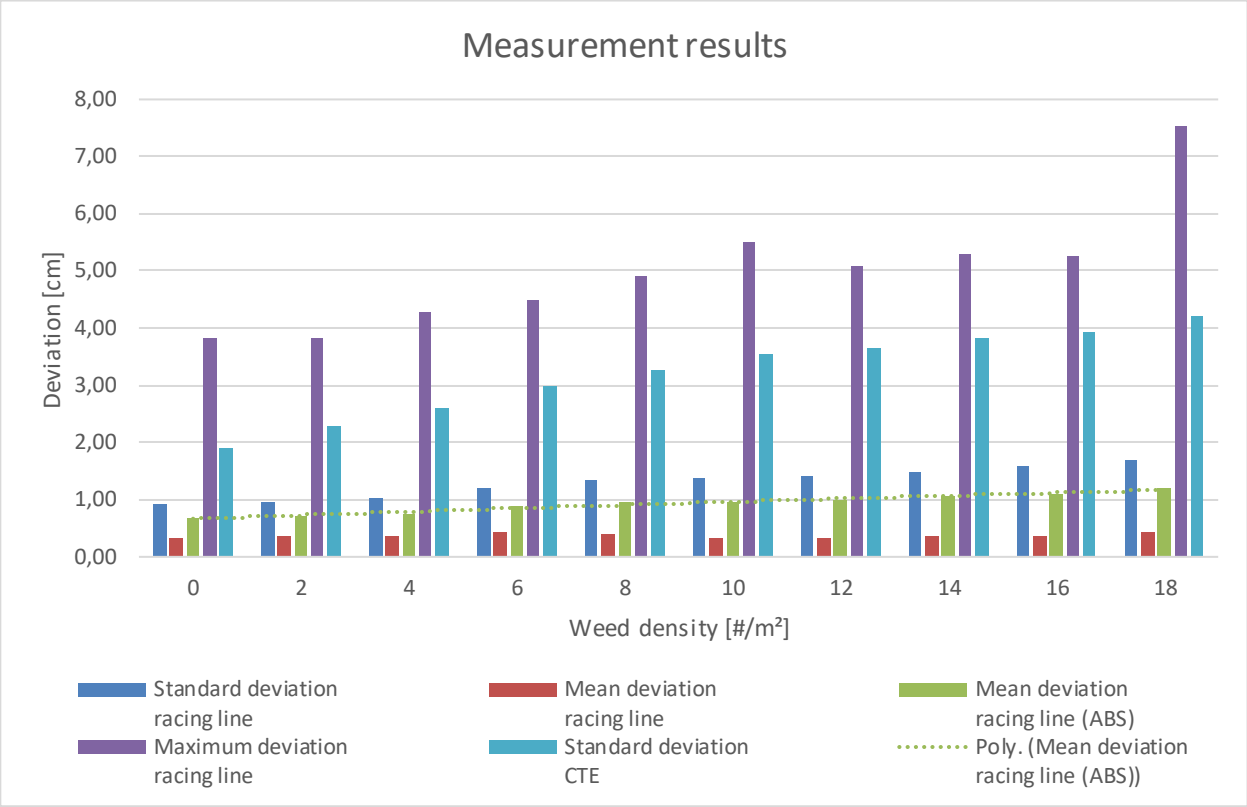


Abbildung 7-46: Messergebnisse mit „Blob reduction“ (50 ± 2 cm Pflanzenabstand)

7.5.3 Messreihen mit 15 cm Pflanzenabstand

7.5.3.1 Beschreibung

Da davon auszugehen ist, dass sich das Messergebnis bei geringerem Pflanzenabstand deutlich verbessert, werden die Messungen aus den Abschnitten 7.5.1 und 7.5.2 mit einem Pflanzenabstand von 15 ± 2 cm wiederholt.

7.5.3.2 Ergebnisse

Im Gegensatz zu Abschnitt 7.5.1 ist der Algorithmus bei einem Pflanzenabstand von 15 ± 2 cm auch ohne „Blob reduction“ imstande, alle Simulationen erfolgreich zu Ende zu führen. Mit einer absoluten mittleren Abweichung von lediglich 7 mm bei einer Unkrautdichte von 18 Stück pro Quadratmeter sind die Ergebnisse sogar deutlich besser als in Abschnitt 7.5.2, bei dem die „Blob reduction“ zum Einsatz kam.

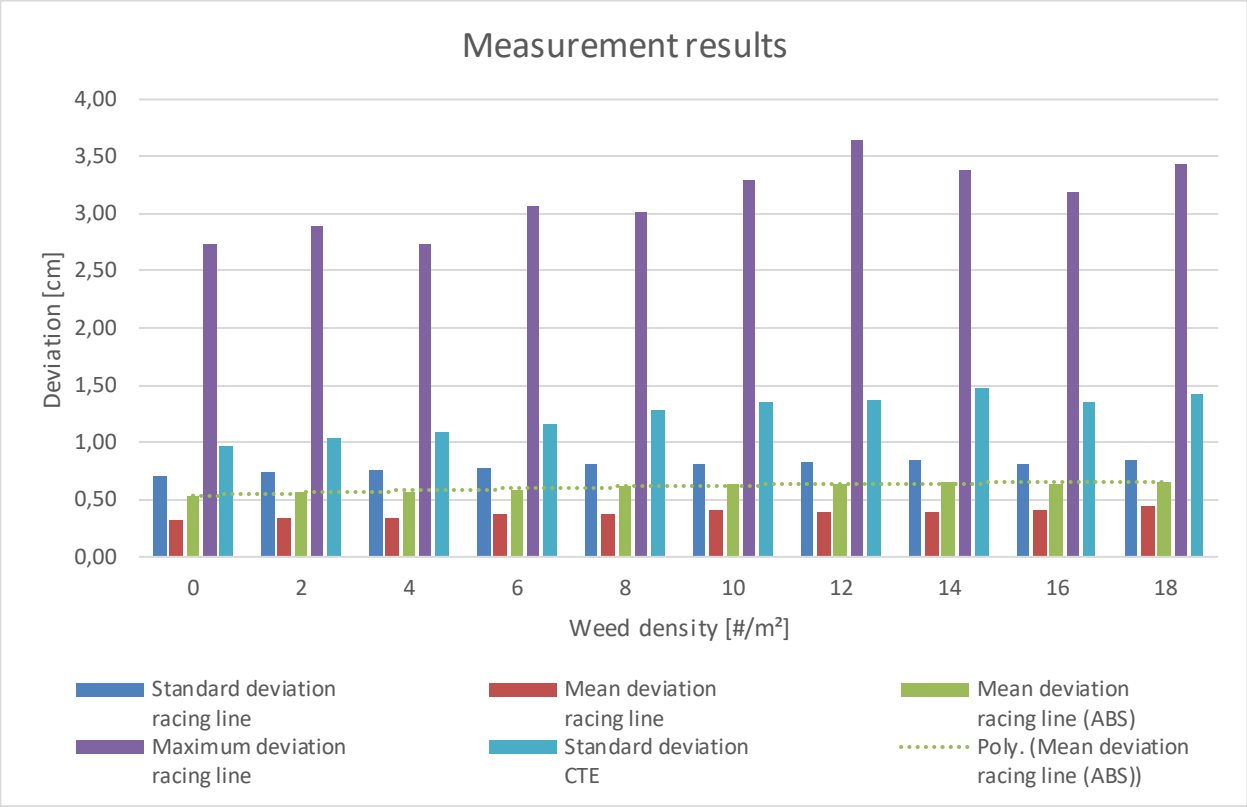


Abbildung 7-47: Messergebnisse ohne „Blob reduction“ (15 ± 2 cm Pflanzenabstand)

Die Messergebnisse der Messreihe mit „Blob reduction“ sind in **Abbildung 7-48** dargestellt. Sie lassen sich aufgrund des bis auf den verringerten Pflanzenabstand unveränderten Messaufbaus mit den Ergebnissen aus **Abbildung 7-46** vergleichen. Gab es in Abschnitt 7.5.3.2 noch eine Abweichung der absoluten mittleren Abweichung von 75% zwischen erster und letzter Messung, so ist diese bei der Simulation mit 15 ± 2 cm Pflanzenabstand und „Blob reduction“ verschwunden. Der verringerte Pflanzenabstand führt dazu, dass sich der Algorithmus nahezu so verhält, als wäre kein Unkraut vorhanden.

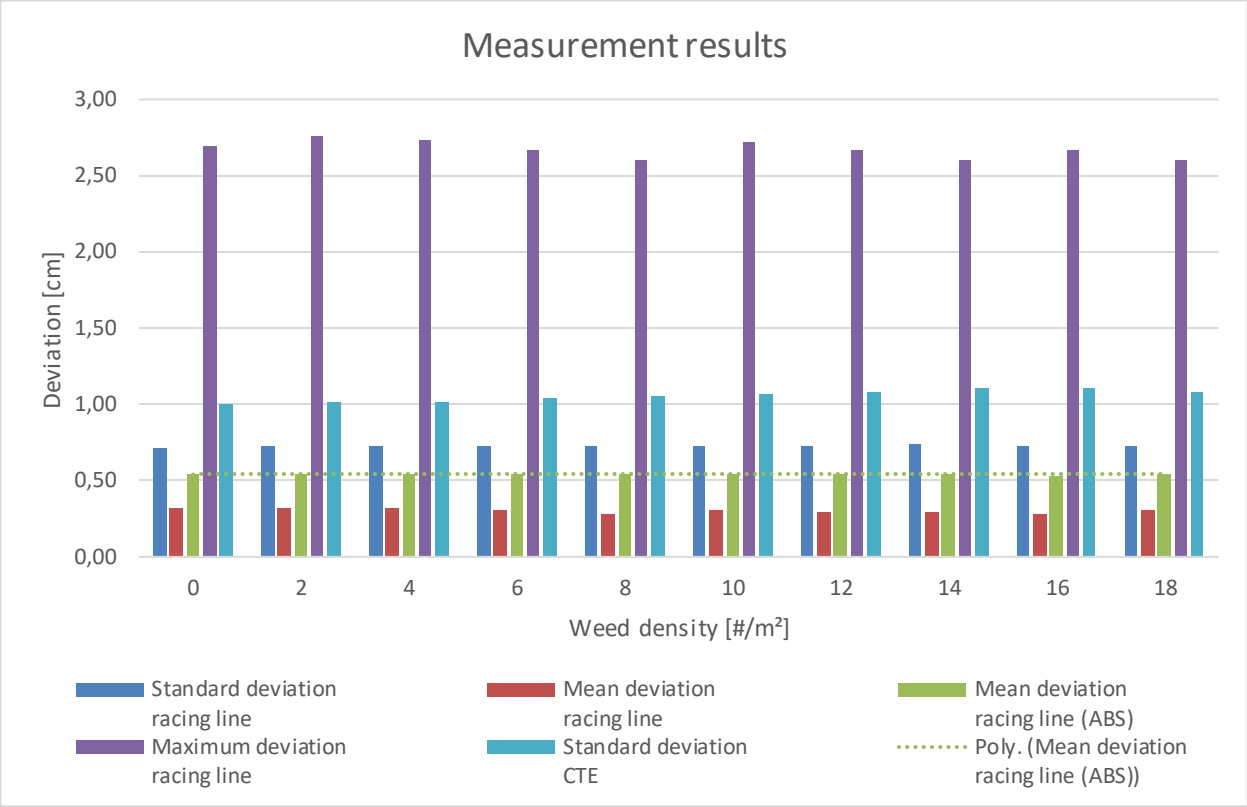


Abbildung 7-48: Messergebnisse mit „Blob reduction“ (15 ± 2 cm Pflanzenabstand)

7.5.4 Analyse Unkraut

In den Abschnitten 7.5.1 bis 7.5.3 konnten einige wertvolle Erkenntnisse gewonnen werden. Es wurde in Abschnitt 7.5.2 gezeigt, dass eine „Blob reduction“ für Kulturpflanzen, die mit größerem Pflanzenabstand gesetzt werden, unabdingbar ist. Die Simulationen aus Abschnitt 7.5.1 konnten mithilfe dieser Erweiterung im Algorithmus allesamt erfolgreich zu Ende geführt werden. Anschließend wurde der Pflanzenabstand auf 15 ± 2 cm reduziert und gezeigt, dass eine „Blob reduction“ auch hier zur Verbesserung der Messergebnisse beitragen kann. Die Voraussetzung für diese Verbesserung ist jedoch die korrekte Parametrisierung des Schwellwerts. Bei realen Messungen empfiehlt es sich, statistische Verfahren anzuwenden, um diesen Wert korrekt einzustellen.

Abgesehen davon, dass die Unkrautdichte auf dem Feld in Wirklichkeit wohl kaum angegeben werden kann, sei für diese Messreihe noch erwähnt, dass das Ergebnis von mehreren Faktoren abhängig ist. Neben der Skalierung der Pflanzen, des Unkrauts und deren Verhältnis zueinander spielen der Pflanzenabstand sowie die Verteilung des Unkrauts eine wesentliche Rolle.

7.6 Robustheit gegenüber Wettereinflüssen

Einer der wohl größten Vorteile der Evaluierung von Algorithmen mittels synthetischer Bilder besteht darin, dass beliebige Wetterszenarien jederzeit simulierbar sind. Die folgenden Messreihen untersuchen das Verhalten des Algorithmus bei unterschiedlichen Wetterbedingungen.

7.6.1 Messreihe Nebel

7.6.1.1 Beschreibung

Zwar ist der Herbst bekanntlich jene Jahreszeit, in der Nebel am häufigsten auftritt, doch können diese fein verteilten Wassertröpfchen, die durch Kondensation von Wasser in der feuchten und übersättigten Luft entstanden sind, ganzjährig in Bodennähe auftreten. Die Simulation von Nebel in der „Unreal Engine“ lässt sich zu jeder Jahreszeit durchführen und bietet neben der Dichte, der Sichtweite und der Albedo, die ein Maß für das Rückstrahlvermögen ist, noch etliche Möglichkeiten, die virtuelle Umgebung so realistisch als möglich zu gestalten. Um möglichst viele unbekannte Variablen auszuschließen, variiert diese Messreihe ausschließlich die „Fog density“ zwischen 0% und 50%. **Abbildung 7-49** zeigt die virtuelle Umgebung mit einer „Fog density“ von 20%. Dabei handelt es sich bereits um einen relativ dichten Nebel; weitere Erhöhungen der Intensität lassen das Bild nicht mehr realistisch wirken.

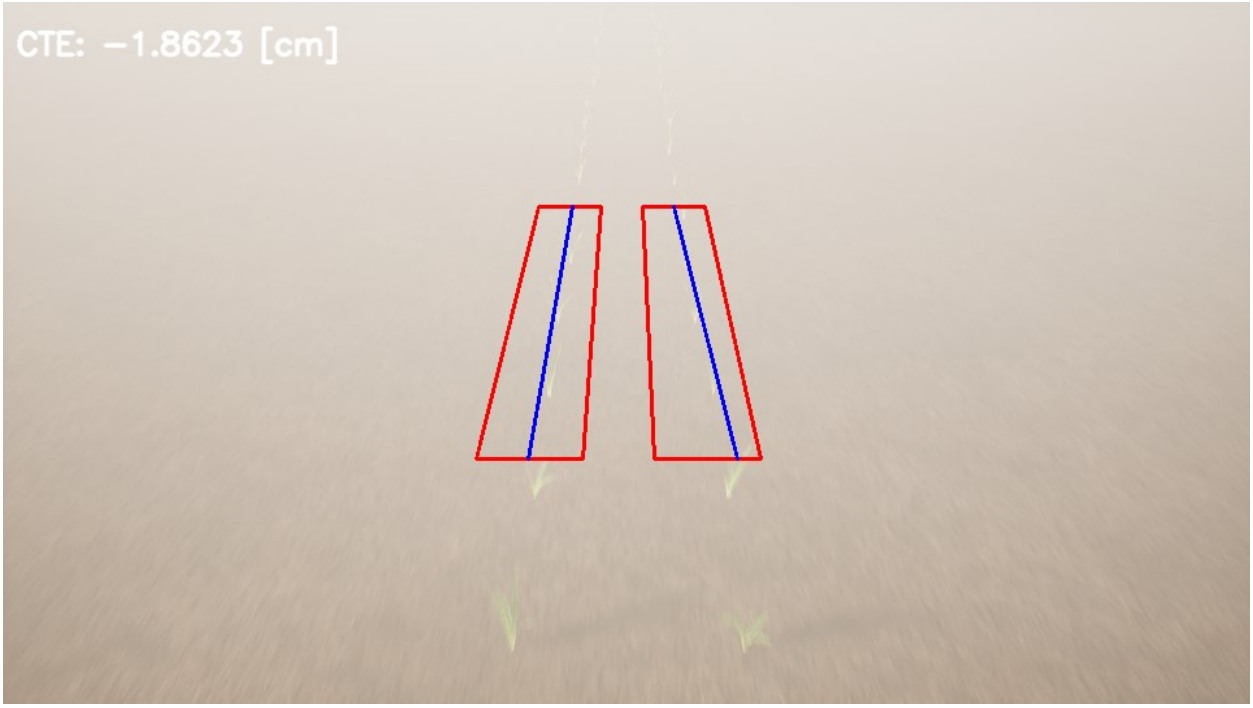


Abbildung 7-49: Virtuelle Umgebung mit einer „Fog density“ von 20%

7.6.1.2 Ergebnisse

Abbildung 7-50 zeigt die Messergebnisse der Simulation. Wie man erkennen kann, ändert sich das Messergebnis bis einschließlich jener Messung, bei der die „Fog density“ 10% erreicht, kaum. Lediglich Schwankungen im Millimeterbereich der maximalen Abweichung sind bemerkbar. Diese sind auf die Totzeiten des Prüfstands zurückzuführen und gleichen sich daher im Mittel wieder aus. Ab einer „Fog density“ von 20% ist ein deutlicher Anstieg der Abweichungen zu verzeichnen.

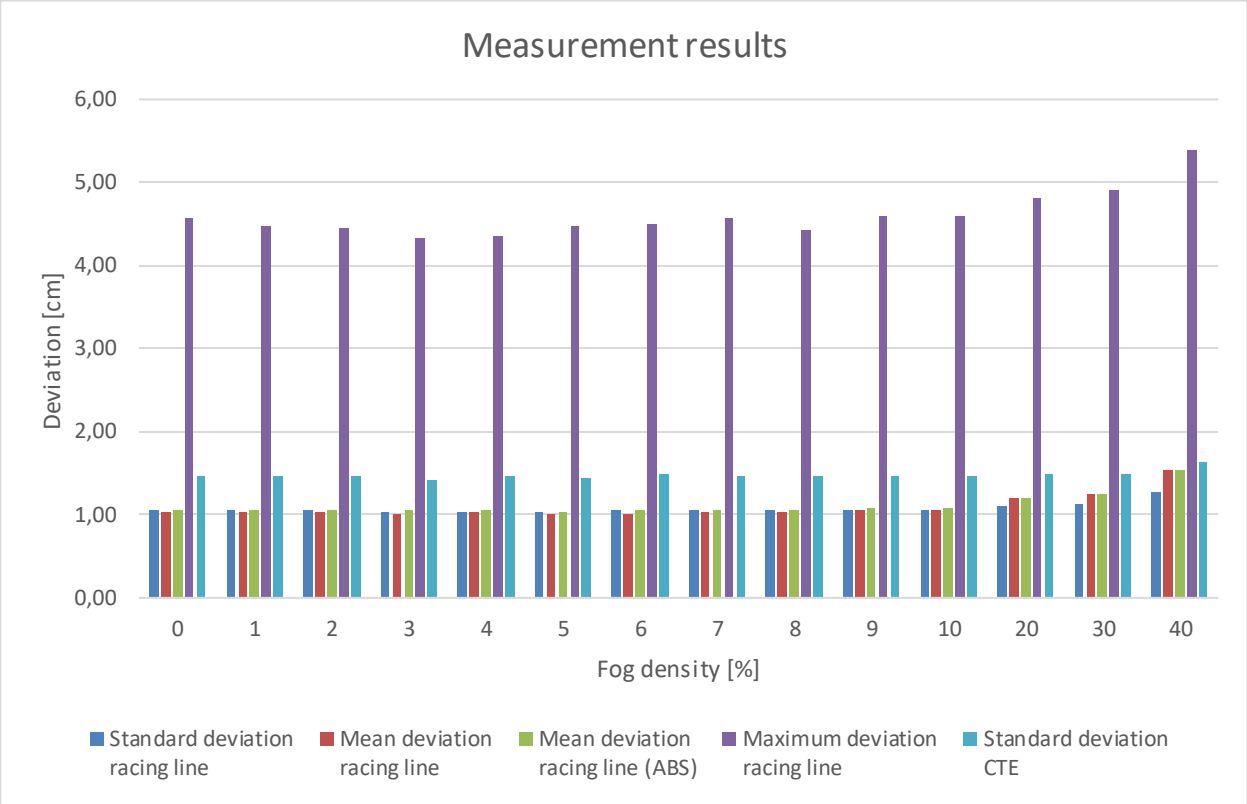


Abbildung 7-50: Messergebnisse Nebel

7.6.2 Messreihe Wasserstände

7.6.2.1 Beschreibung

Nach längerem Regen kann es in den Furchen des Ackers zu einer Pfützenbildung kommen. Diese Wasserstände können mithilfe des „Substances Plugin“ simuliert werden. **Abbildung 7-51** zeigt eine Simulation mit einem Wasserstand von 20%. Zwar können die Wasserstände bis zu 100% angehoben werden, doch wirken Bilder mit über 40% Wasserstand teilweise unrealistisch.

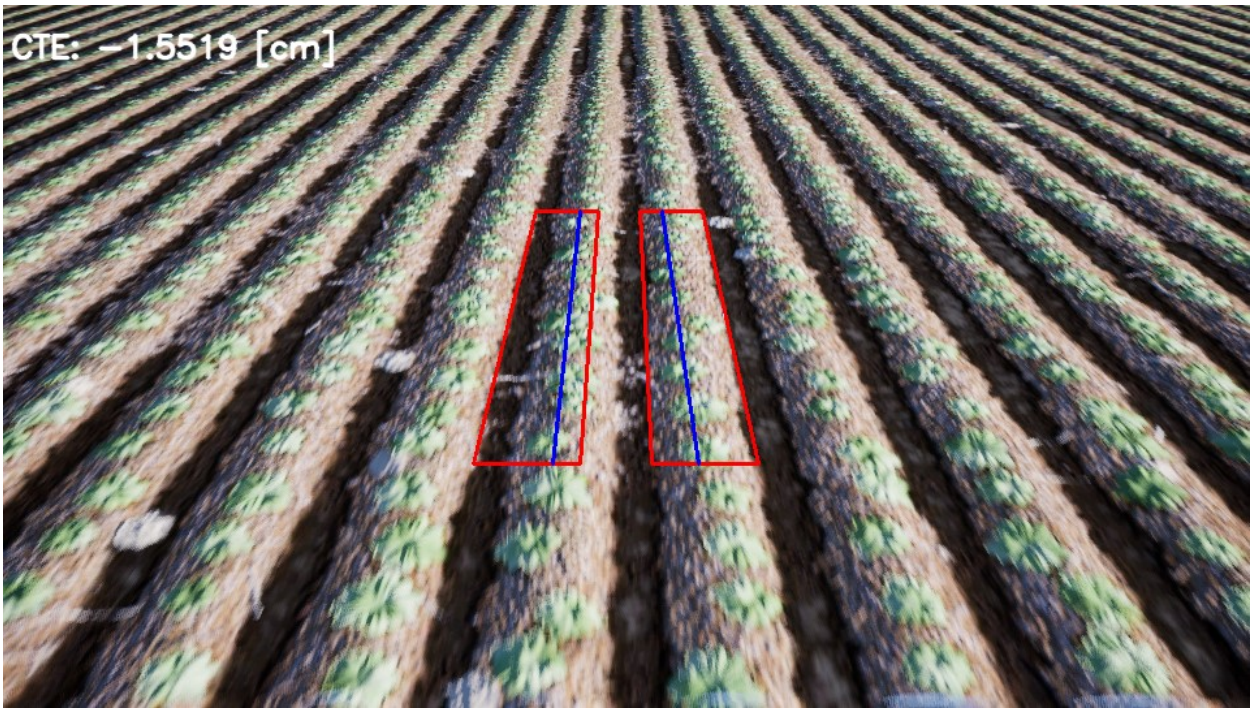


Abbildung 7-51: Virtuelle Umgebung Wasserstand 20%

7.6.2.2 Ergebnisse

Wie auch schon in Abschnitt 7.3.3 ist die Position der Pflanzen nicht bekannt, weshalb in **Abbildung 7-52** nur die Standardabweichung und die maximale Abweichung des „Cross Track Errors“ dargestellt werden können. Die Standardabweichung des CTE nimmt ab 20% leicht zu, was daran liegt, dass ab diesem Wert die Pflanzen teilweise unter dem Wasser verschwinden und für die Reihendetektion daher weniger Pflanzen zur Verfügung stehen.

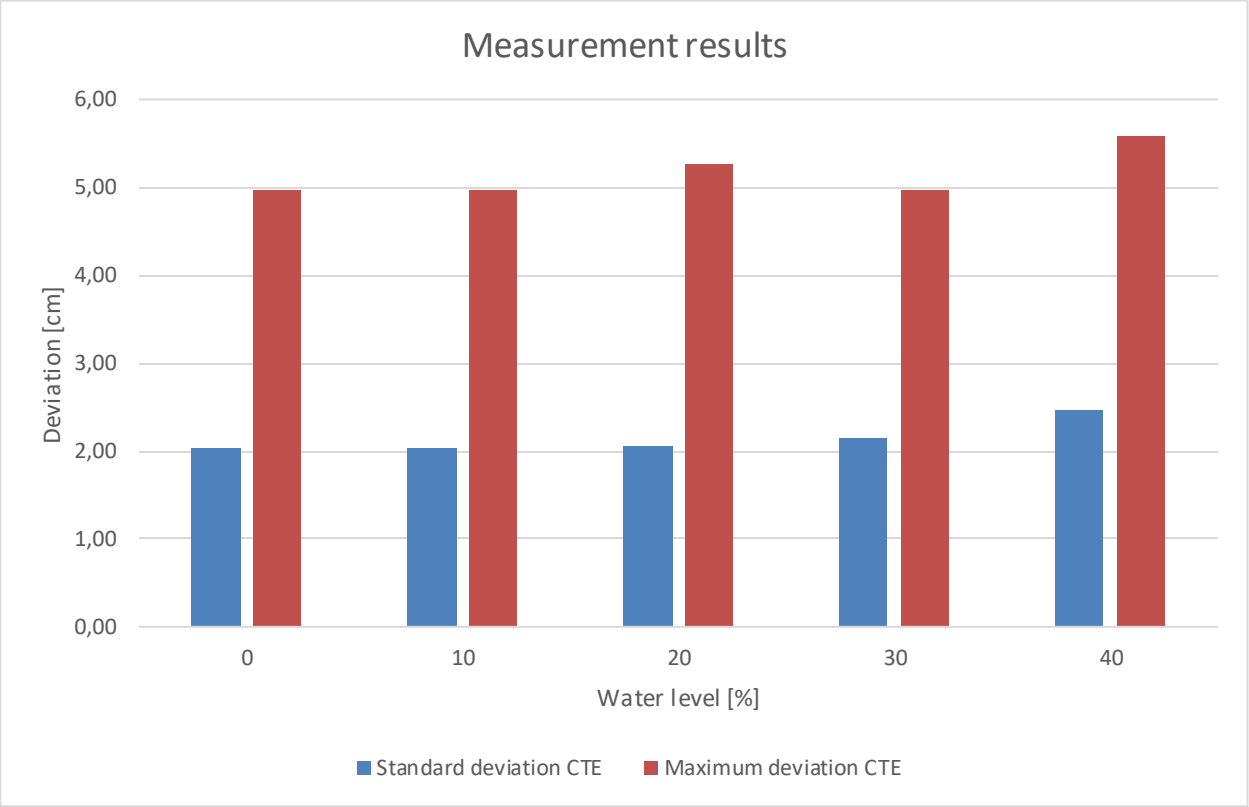


Abbildung 7-52: Messergebnisse Wasserstände

7.6.3 Analyse der Wettereinflüsse

In den Abschnitten 7.6.1 und 7.6.2. wurden Messreihen mit Nebel und Wasserpfützen simuliert und dabei festgestellt, dass (sofern die Simulation realistische Bilder liefert) der Algorithmus weitgehend stabil gegen Wettereinflüsse ist. Grund dafür ist, dass Wettereinflüsse wie beispielsweise der Nebel ausschließlich Auswirkungen auf die Farbdarstellung der Pflanzen haben. Sofern diese Farbdarstellungen keine unnatürlichen Ausmaße annehmen, ist der Algorithmus mit dem „Excess green index“ in der Lage die Simulation erfolgreich abzuschließen (vgl. Abschnitt 7.3).

8 ANALYSE DER SIMULATIONS - ERGEBNISSE

In den Abschnitten 7.1 bis 7.6 wurde stets die wissenschaftlichen Arbeitsweise eingehalten und Variablen in den einzelnen Messreihen nur systematisch geändert. Nach 17 durchgeführten Messreihen haben sich dadurch gewisse Tendenzen abgezeichnet, welche nachfolgend erläutert werden sollen.

Die grundlegenden Simulationen aus Abschnitt 7.1 zeigen, dass eine realitätsnahe Messung bei welcher ein Pflanzenabstand von 15 ± 2 cm und eine Rotation der Pflanze zwischen 0° und 360° simuliert wird, eine absolute mittlere Abweichung von 5 mm und eine Standardabweichung (SD) von 7 mm hat. Erhöht man den Pflanzenabstand auf 50 ± 2 cm, was in etwa einer Kürbiskultur entspricht, verschlechtert sich dieses Ergebnis auf 11 mm mittlere Abweichung und eine SD von 11 mm. Während das Aussetzen von Pflanzen bei den Simulationen mit 15 ± 2 cm kaum eine Rolle spielt, ist das Ergebnis aus **Tabelle 7-2** mit einer Ausfallsrate des Algorithmus von 60% unbefriedigend.

Die mittlere Abweichung des Algorithmus ändert sich durch die Simulation von unterschiedlichen Pflanzen um teilweise mehr als 10 mm, was als eindeutiger Nachteil des getesteten Algorithmus identifiziert werden kann.

Durch die Messreihen in Abschnitt 7.2 konnte gezeigt werden, dass Kameraeffekte einen positiven Einfluss auf das Simulationsergebnis haben. Die Ergebnisse verbesserten sich durch zunehmende Kameraeffekte um ungefähr 20%. Weiters zeichnen sich hier die ersten Vorteile eines „Software in the Loop“ Prüfstandes ab. Durch die Simulation der Kamera lassen sich Effekte wie die Bewegungsunschärfe gezielt variieren, was es schlussendlich möglich macht, deren Auswirkungen zu untersuchen.

Die Robustheit gegenüber Farbabweichungen ist ein entscheidendes Kriterium für Algorithmen, bei welchen die Pflanzenreihenerkennung auf dem „Excess green index“ basiert. Es konnte in Abschnitt 7.3 gezeigt werden, dass der Ansatz mittels ExG erstaunlich gute Ergebnisse liefert. So unterscheidet sich die absolute mittlere Abweichung bei den Messungen mit 3000 K und 6500 K lediglich um 2 mm. Ähnliche Ergebnisse weisen auch die Messreihen mit unterschiedlichen Böden auf. Kein gutes Ergebniss konnte jedoch bei der Simulation zusätzlicher Grasflächen erreicht werden. Vor- und Nachteile der künstlichen Farbdarstellung liegen nahe beieinander. Einerseits lassen sich die Farbwerte jedes einzelnen Pixels bei synthetischen Bildern beeinflussen, doch andererseits sind die Darstellungen von Farben in der Natur dermaßen vielfältig, dass sie nie in einer Simulation wiedergegeben werden können.

Die größten Abweichungen zwischen gefahrener Linie und Ideallinie finden sich in Abschnitt 7.4.3. Der Grund dafür liegt in der fehlenden Rückführung der Eulerwinkel. In der Simulation lässt sich diese Rückführung einfach bewerkstelligen. In der Praxis ist für die Berechnung dieser Winkel ein Gyroskop notwendig. Ein Kamerasystem, welches ein Gyroskop und einen

Beschleunigungssensor beinhaltet, wird von Intel unter dem Namen Intel Realsense D435i („inertial measurement unit“) angeboten.

Zwar wurde in dieser Arbeit bereits mehrmals erwähnt, dass sich der zu untersuchende Algorithmus nicht nur für das automatisierte Unkrautjäten eignet, doch wurde er ursprünglich für diesen Zweck entwickelt. Aus diesem Grund ist es nur günstig, dass die Simulationsergebnisse aus Abschnitt 7.5 auf eine beinahe Invarianz des Algorithmus auf Unkraut hindeuten.

Die Simulation von Witterungsbedingungen wurde bereits in der Einleitung als Vorteil eines SiL Prüftands beworben. Umso überraschender sind die Messergebnisse aus Abschnitt 7.6. Die Abweichungen steigen weder mit zunehmendem Nebel noch mit zunehmendem Wasserstand erheblich an. Da die Darstellung in **Abbildung 7-49** aufgrund des volumetrischen Nebels jedoch realistisch wirkt, ist davon auszugehen, dass derartige Ergebnisse auch in der Realität erzielt werden können.

Zusammenfassend kann gesagt werden, dass in den vorangegangen Abschnitten alles darangesetzt wurde, möglichst realitätsnahe Szenarien für die Evaluierung des Algorithmus zu erstellen. Bis auf wenige Ausnahmen war der Algorithmus in der Lage, diese Simulationen ohne größere Abweichungen zu absolvieren. Neben der eigentlichen Evaluierung des Algorithmus konnte noch gezeigt werden, dass sich sowohl die „Lineare Regression“ als auch der „Excess green index“ gut für die Detektion von Pflanzenreihen eignen.

9 VERGLEICH MIT REALEN MESSUNGEN

9.1 Reale Bilder

Um zu evaluieren wie 'gut' die synthetischen Bilder im Vergleich zu realen Bildern sind, wurden Bilder auf einem Maisfeld aufgenommen und mit jenen aus der „Unreal Engine" verglichen. Da sich die Szene in der UE nicht 1:1 nachstellen lässt, werden ähnlich wie in den vorangegangenen Abschnitten, Kameraparameter an der Intel Realsense D435 variiert und die Auswirkungen verglichen. **Abbildung 9-1** zeigt den Messaufbau am Maisfeld. Die Kamera wurde 160 cm über dem Boden in einem Winkel von -30° montiert, was in etwa auch dem Aufbau am Implement entspricht. Es wurden 39 Bilder aufgenommen, welche anschließend mit „OpenCV" und dem „Digital Plant Analyzer" analysiert wurden.

Abbildung 9-1: Messaufbau am Maisfeld

Abbildung 9-2 steht exemplarisch für eine der 39 aufgenommenen Messungen. Sie zeigt eine Aufnahme der Intel Realsense mit eingezeichneter „Region of Interest". Diese wird vergrößert und anschließend manuell eine Linie eingezeichnet, welche die Pflanzenreihe approximieren soll.

Die eingezeichnete Linie wird schlussendlich mit den Ergebnissen der „Linearen Regression“ und der „Stripe Analysis“ verglichen. Wie man sehen kann, erzielen beide Methoden auch bei realen Bildern durchaus zufriedenstellende Ergebnisse. Um diese zu untermauern, müssten jedoch dutzende Bilder von Experten analysiert werden, da eine minimale Veränderung der manuell eingezeichneten Linie bereits große Auswirkungen auf das Ergebnis hat.

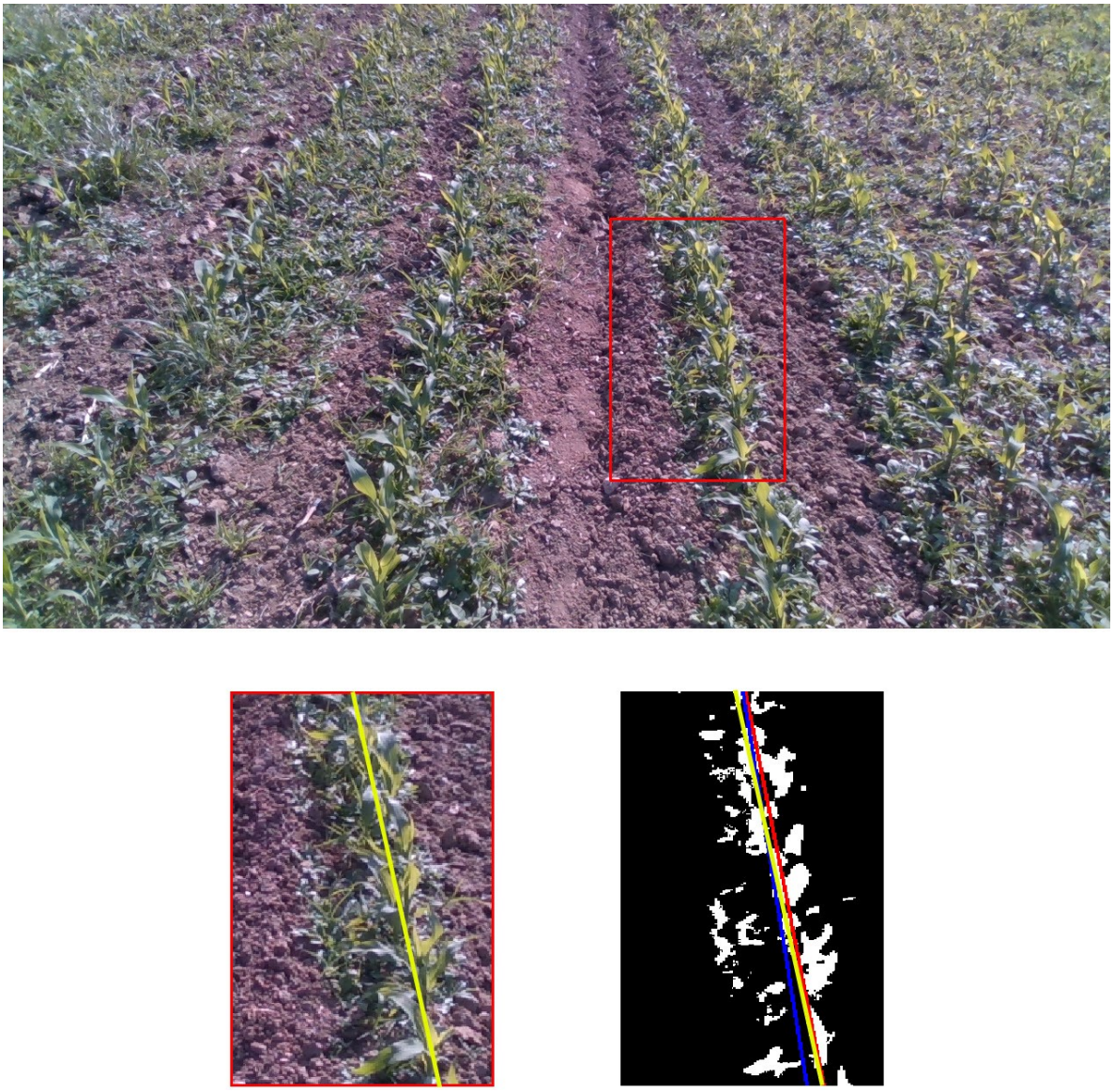


Abbildung 9-2: Feldaufnahme, subjektive Einschätzung (Gelb), „Lineare Regression“ (Blau), „Stripe Analysis“ (Rot)

9.2 Auswertung der Bilder

Abbildung 9-3 zeigt das Grauwerthistogramm der am Feld erstellten Aufnahme. Es wird aus dem „value“ Kanal des HSV-Farbraums („hue, saturation, value“) berechnet und stellt die Häufigkeit der vorkommenden Grauwerte grafisch dar. Mithilfe von derartigen Histogrammen kann festgestellt werden, ob Bilder unter- beziehungsweise überbelichtet sind. Bei einem unterbelichteten Bild werden die meisten Grauwerte im linken Teil des Histogramms zu finden sein und bei einem überbelichteten Bild vorwiegend im rechten Teil [63, Ch. 5.1]. Aus **Abbildung 9-3** lässt sich daher ableiten, dass das am Feld aufgenommene Bild kontrastreich und weder unter- noch überbelichtet ist.

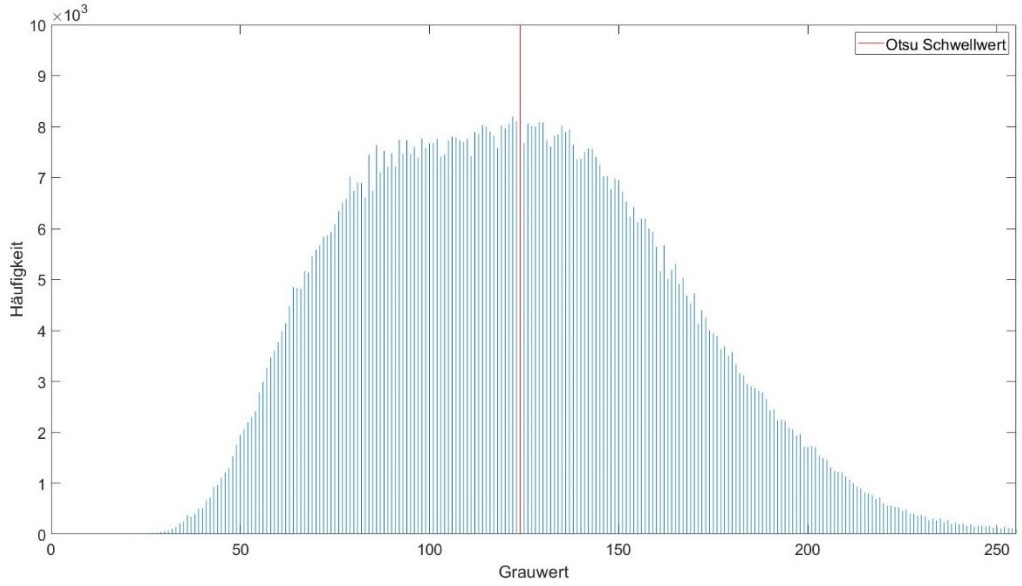


Abbildung 9-3: Histogramm von Abbildung 9-2 (oben)

Der Werteberich des „value“ Kanals liegt zwischen 0 und 1 und wird für die Visualisierungen und die Berechnung der Histogramme auf den Bereich [0,255] skaliert. Theoretisch wäre es bereits jetzt möglich, mittels Otsus Schwellwertmethode, ein Binärbild zu erstellen. Bei dem Verfahren nach Otsu handelt es sich um eine Schwellwertmethode, welche die Streuung der Grauwerte beurteilt. Um den Schwellwert festzulegen, wird jedes Pixel der Szene analysiert und kategorisiert, ob es sich dabei um eine Pflanze oder um Boden handelt [9, Ch. 4.3]. Mit dem aus **Abbildung 9-4** erstellten Binärbild wäre eine korrekte Detektion der Pflanzenreihe allerdings noch nicht möglich. Um diese zu gewährleisten, wurde die die „Region of Interest“ (ROI) und der „Excess green index“ (ExG) eingeführt. Die ROI dient dazu, dass ausschließlich für die Reihenerkennung relevante Bildausschnitte im Algorithmus weitergegeben werden.

Abbildung 9-4: Grauwertbild aus Abbildung 9-2 (oben)

Die Anwendung des ExG auf einen dieser relevanten Bildausschnitte liefert ein Grauwertbild, das in **Abbildung 9-5** inklusive Histogramm dargestellt ist.

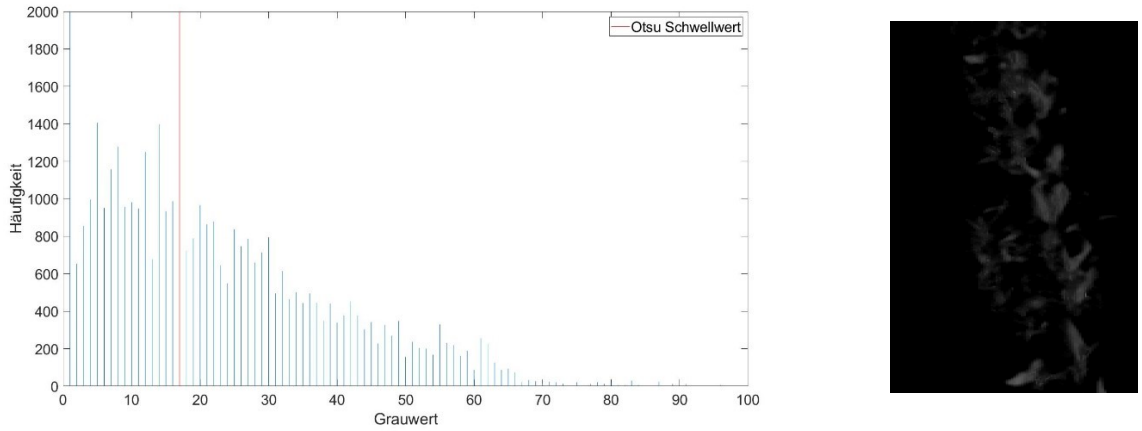


Abbildung 9 5: Histogramm und Grauwertbild einer ROI

Die Festlegung einer ROI und die Anwendung des ExG haben zur Folge, dass das Histogramm deutlich schmäler wird und sich der Otsu Schwellwert verändert.

Stellt man abschließend jene Pixel, welche unter dem Schwellwert liegen in weiß dar und jene welche den Wert überschreiten schwarz; erhält man ein Binärbild. Dieses Binärbild kann genutzt werden, um mit Hilfe der „Linearer Regression“ oder „Stripe Analysis“ die Pflanzenreihen zu detektieren.

9.3 Synthetische Bilder

Eine der Fragen, welche durch die vorliegende Arbeit beantwortet werden soll lautet: „Wie gut lassen sich bildgestützte Algorithmen mit synthetischen Bildern evaluieren?“ Um diese Frage beantworten zu können, müssen sowohl reale als auch synthetische Bilder analysiert werden. T. Vaudrey et al. [64] stellten fest, dass einer der größten Unterschiede zwischen realen und synthetischen Bildern die offensichtlichen Objektgrenzen (Kanten) sind. Diese sind in synthetischen Bildern deutlich stärker ausgeprägt, was zu schnellen Änderungen der Grauwerte und damit zum steileren Anstieg der Gradienten führt. Sowohl Algorithmen welche auf Stereobildern basieren als auch Algorithmen welche den optischen Fluss bestimmen, kommt diese steile Änderung des Gradienten zugute [64]. Ein weitere Unterschied ist, dass sich die Intensität zwischen den einzelnen Bildern bei synthetischen Bildern im Vergleich zu realen Bildern nicht ändert [64]. Unter den „Camera Settings“ bietet die „Unreal Engine“ auch für dieses Problem Einstellmöglichkeiten. Wie schnell die Kamera auf sich ändernde Lichtverhältnisse reagieren soll lässt sich mithilfe der Parameter Speed up und Speed down festlegen. Speed up beschreibt hierbei die Geschwindigkeit mit welcher die Anpassung von einer dunklen Umgebung an eine helle erfolgen soll und Speed down jene von einer hellen an eine dunkle Umgebung [65].

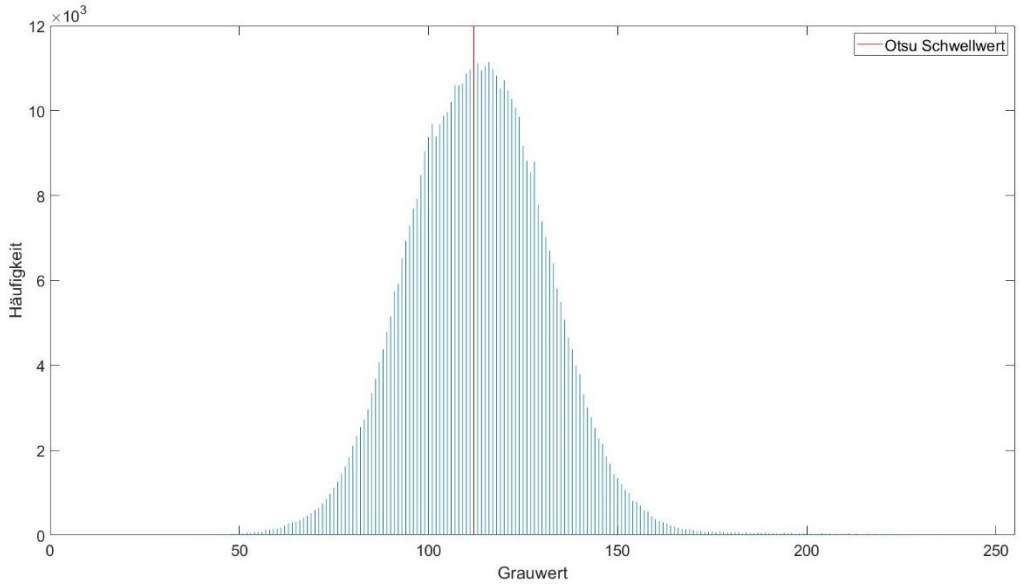


Abbildung 9-5: Histogramm eines generierten Bildes (komplette Szene)

Abbildung 9-5 zeigt ein Histogramm, das ebenfalls aus dem „value“ Kanal des HSV-Farbraums erstellt wurde. Typisch für die in der vorliegenden Arbeit generierten Bilder ist, dass sie im Vergleich zu realen Bildern deutlich kontrastärmer sind. Das lässt sich auf die fehlende Vielfalt der künstlich erstellten Bilder zurückführen. Dass eine Analyse der kompletten Szene auch bei synthetischen Bildern nur bedingt möglich ist, zeigt **Abbildung 9-6**. In dieser Abbildung wurde vorab das Grauwertbild berechnet und anschließen mit Hilfe der Otsu Schwellwertmethode das

Binärbild erstellt. Wie zu erkennen ist, ist eine Detektion der Pflanzen aus dem Binärbild noch nicht möglich. Daraus lässt sich schließen, dass der Einsatz einer ROI und des ExG, sowohl für reale als auch für synthetische Bilder, unerlässlich für die in dieser Arbeit vorgestellte Methode ist.

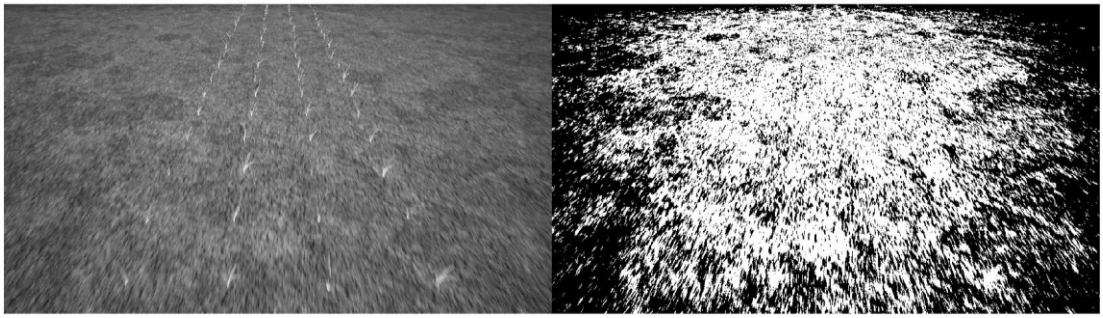


Abbildung 9-6: Grauwertbild (links), Binärbild (rechts)

9.4 Erkenntnisse

Neben der **Abbildung 9-2** wurden noch 38 weitere reale Bilder ausgewertet und die Ergebnisse zeigen, dass der Algorithmus Probleme mit

- überbelichteten Bildern
- ungesättigten Bildern und
- Bildern bei welchen der Farbwert (Hue) stark abweicht

hat. Diese Erkenntnis deckt sich auch mit jener, welche aus den synthetischen Bildern gewonnen wurde. Dennoch konnte gezeigt werden, dass Aufnahmen, welche mit automatischer Belichtung gemacht werden, ausreichend sind, um Pflanzenreihen mittels Algorithmus zu detektieren. Ein ausschlaggebender Faktor für die erfolgreiche Detektion von Pflanzenreihen ist die „Region of Interest“. Würde man stets die komplette Szene analysieren, kommt es neben dem erhöhten Berechnungsaufwand auch zu Fehldetektionen.

Die Abweichung zwischen detektierter und manuell eingezeichneter Pflanzenreihe kann als Maß für die Genauigkeit herangezogen werden. Voraussetzung dafür ist jedoch ein deutlich größerer Datensatz an realen Bildern mit manuell annotierten „Ground Truth“ Daten.

10 ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit wurde ein „Software in the Loop“ (SiL) Prüfstand entwickelt mit dessen Hilfe ein bereits vorhandener Algorithmus im Detail evaluiert wurde. Als Basis für die virtuelle Umgebung diente die „Unreal Engine“, eine der aktuell leistungsstärksten Spiele Engines. In Kombination mit dem „Robot Operating System“ ließ sich damit ein leistungsstarker Prüfstand entwickeln, welcher auch für die Evaluierung anderer bildgestützter Algorithmen eingesetzt werden kann. Mittels C++ API lassen sich damit komplexe Prüfszenarien erstellen und Ergebnisse fundiert evaluieren. Um die Reproduzierbarkeit der Messergebnisse zu gewährleisten wurde das SiL Konzept um einen Datenspeicher erweitert. Dieser beinhaltet sämtliche Informationen zum Testsetup, die „Ground Truth“ Daten, sowie Simulationsergebnisse.

Die Simulationsergebnisse wurden ausführlich evaluiert und festgestellt, dass der Algorithmus auch anspruchsvolle Simulationen, mit variierenden Licht- und Wetterverhältnissen, starkem Unkrautbewuchs und parasitäre Kameraeffekte ohne große Abweichungen bewältigen kann. Raum für Verbesserung bleibt hingegen bei Simulationen, bei der sich die Eulerwinkel über die Distanz ändern. Das betrifft neben der Kameraerschütterungen auch Simulationen, bei denen eine Kurve durchfahren werden soll. Wie auch schon in der vorangegangen Arbeit, in der dieser Algorithmus entwickelt wurde, konnten mit der „Linearen Regression“ bessere Ergebnisse als mit der „Stripe Analysis“ erzielt werden.

Als große Vorteile des SiL Konzepts haben sich die Möglichkeit beliebige Szenarien ganzjährig zu simulieren, die Verfügbarkeit sämtlicher Daten und die Reproduzierbarkeit der Messergebnisse herausgestellt. Nachteilig soll der hohe Aufwand, der für die Erstellung einer realitätsnahen virtuellen Umgebung nötig ist, erwähnt werden. Die fehlende Vielfalt der Pflanzen ist als klarer Schwachpunkt dieser Arbeit auszumachen. Unter anderem wurden auch bei dem Boden und der virtuellen Kamera, Abstriche in Kauf genommen. Diese Einschränkungen könnten Thema in weiterführenden Arbeiten sein.

Die synthetischen Bilder, die in dieser Arbeit generiert worden sind, sind von Fotorealismus noch weit entfernt, doch wird in den vorangegangenen Kapiteln gezeigt, welche Möglichkeiten sich für die Evaluierung von bildgestützten Algorithmen bieten. Neben dieser Evaluierung soll auch noch die Möglichkeit zur Validierung mathematischer beziehungsweise statistischer Methoden mittels SiL Konzept erwähnt werden. So ermöglicht es das ROS, dass sämtliche Programme, welche eine Programmierschnittstelle enthalten miteinander kommunizieren können. Zusammen mit der Verfügbarkeit sämtlicher Daten, angefangen vom Farbwert eines Pixels, bis hin zum komplexen Ergebnis einer Berechnung, bietet das unzählige Möglichkeiten.

Diese Arbeit konnte zeigen, dass die „Unreal Engine“ mit ihrem fotorealistischen Rendering ausreichenden Realismus schaffen kann, um bildgestützte Algorithmen für die Landwirtschaft wissenschaftlich zu evaluieren.

11 ABBILDUNGSVERZEICHNIS

Abbildung 1-1: Kerntechnologien der automatischen Unkrautbekämpfung 7
Abbildung 3-1: Vor- und Nachteile alternativer Lösungswege 11
Abbildung 4-1: Funktionsprinzip „Prüfstand“ 15
Abbildung 4-2: Flussdiagramm des Algorithmus zur Reihendetektion 16
Abbildung 4-3: ROS Framework SiL-Prüfstand 18
Abbildung 4-4: „Stripe Analysis“ für ein 9x9 Binärbild 23
Abbildung 4-5: Berechnung der Ideallinie 25
Abbildung 4-6: Messreihe zum Feststellen der Präzision 27
Abbildung 4-7: (links) CTE einer parallelen Ideallinie; (rechts) CTE einer schrägen Ideallinie 28
Abbildung 5-1: Funktionsprinzip „Blueprints“ mit 2 „nodes“ 30
Abbildung 5-2: Pflanze und Unkraut an derselben Position 31
Abbildung 5-3: Virtuelle Umgebung 34
Abbildung 6-1: Position der Kamera auf dem Feld (modifizierte Abbildung aus [46], Farbschema „Unreal Engine“) 38
Abbildung 7-1: Messergebnisse „Lineare Regression“ vs. „Stripe Analysis“ 43
Abbildung 7-2: Messergebnis mit „Linearer Regression“ 43
Abbildung 7-3: Messergebnisse „Linearer Regression“ vs. „Stripe Analysis“ mit Offset 44
Abbildung 7-4: Simulation Aussetzer in der Pflanzenreihe 50 ± 2 cm Pflanzenabstand (LR) 46
Abbildung 7-5: Pflanze 1 (Links), Pflanze 2 (Mitte), Pflanze 3 (Rechts) 48
Abbildung 7-6: Analyse der LR (Blau) und SA (Rot), Ideallinie (Grün) 52
Abbildung 7-7: Kompletter Datensatz (Weiß), reduzierter Datensatz (Rot), „Ground Truth“ Daten (Grün) 53
Abbildung 7-8: Eingefärbtes Binärbild einer Pflanze (links ohne „Blob reduction“, rechts mit) 53
Abbildung 7-9: Virtuelle Umgebung ohne Kameraeffekte 54
Abbildung 7-10: Virtuelle Umgebung mit einem „Motion Blur Amount“ von 100% 55
Abbildung 7-11: Messergebnisse Bewegungsunschärfe 56
Abbildung 7-12: Messergebnisse Bildrauschen 57

Abbildung 7-13: Virtuelle Umgebung Textur: „T_Metal_Rust_D“ (100%) 58

Abbildung 7-14: Binärbild generiert mit Otsu Schwellwertmethode aus Abbildung 7-10 60

Abbildung 7-15: Binärbild manueller Schwellwert 20, generiert aus Abbildung 7-13 61

Abbildung 7-16: Ergebnisse Messreihe 1: Boden 1 – Pflanze 1 62

Abbildung 7-17: Farbsättigung der Pflanzen von links nach rechts, saturation: 42%, 50%, 100% 65

Abbildung 7-18: Messergebnisse Farbsättigung der Pflanze 66

Abbildung 7-19: Virtuelle Umgebung mit zusätzlichen Grasflächen; „Ground grass amount“: 20% 67

Abbildung 7-20: Messergebnisse zusätzliche Grasflächen 68

Abbildung 7-21: Virtuelle Umgebung am 1. Mai zu unterschiedlichen Tageszeiten (von rechts nach links solar time: 6 Uhr, 12 Uhr und 20 Uhr) 69

Abbildung 7-22: Messergebnisse simuliertes Datum, oben 1. Mai unten 15. Mai 70

Abbildung 7-23: Messergebnisse simuliertes Datum, oben 1. Juni unten 15. Juni 71

Abbildung 7-24: Virtuelle Umgebung mit Penumbra, Solar time: 8 Uhr 72

Abbildung 7-25: Zugfahrzeug 3D-Darstellung 73

Abbildung 7-26: Zugfahrzeug Drahtgitter-Modell 73

Abbildung 7-27: Messergebnisse Penumbra 1. Juni 74

Abbildung 7-28: ROI nach Otsu Schwellwertmethode 1. Mai von 6 Uhr bis 20 Uhr 75

Abbildung 7-29: Binärbilder 1. Mai von 6 bis 20 Uhr 76

Abbildung 7-30: Bitmap derselben Pflanze rot um 7 Uhr und grün um 12 Uhr 76

Abbildung 7-31: Messergebnisse Kameraerschütterungen 78

Abbildung 7-32: Rotation 0° 79

Abbildung 7-33: Rotation 180° 79

Abbildung 7-34: Messergebnisse Skalierung der Pflanzen, Rotation 0° 80

Abbildung 7-35: Messergebnisse Skalierung der Pflanzen, Rotation 180° 81

Abbildung 7-36: Messergebnisse Variation Gierwinkel 82

Abbildung 7-37: Messergebnisse Variation Nickwinkel 83

Abbildung 7-38: Messergebnisse Variation Rollwinkel (50 cm Pflanzenabstand oben, 15 cm unten) 84

Abbildung 7-39: Simulation Kurvenfahrt 85

Abbildung 7-40: Berechneter Schwerpunkt einer Pflanze 86

Abbildung 7-41: Binärbilder mit unterschiedlichen Skalierungen der Pflanzen (rot detektierte Linie grün Ideallinie) 87

Abbildung 7-42: Variation Gierwinkel 88

Abbildung 7-43: Variation Rollwinkel 88

Abbildung 7-44: Virtuelle Umgebung mit einer Unkrautdichte von 14 Stück/m^2 89

Abbildung 7-45: Simulation ohne „Blob reduction“ (50 ± 2 cm Pflanzenabstand) 90

Abbildung 7-46: Messergebnisse mit „Blob reduction“ (50 ± 2 cm Pflanzenabstand) 91

Abbildung 7-47: Messergebnisse ohne „Blob reduction“ (15 ± 2 cm Pflanzenabstand) 92

Abbildung 7-48: Messergebnisse mit „Blob reduction“ (15 ± 2 cm Pflanzenabstand) 93

Abbildung 7-49: Virtuelle Umgebung mit einer „Fog density“ von 20% 95

Abbildung 7-50: Messergebnisse Nebel 96

Abbildung 7-51: Virtuelle Umgebung Wasserstand 20% 97

Abbildung 7-52: Messergebnisse Wasserstände 98

Abbildung 9-1: Messaufbau am Maisfeld 101

Abbildung 9-2: Feldaufnahme, subjektive Einschätzung (Gelb), „Lineare Regression“ (Blau), „Stripe Analysis“ (Rot) 102

Abbildung 9-3: Histogramm von Abbildung 9-2 (oben) 103

Abbildung 9-4: Grauwertbild aus Abbildung 9-2 (oben) 104

Abbildung 9-5: Histogramm eines generierten Bildes (komplette Szene) 105

Abbildung 9-6: Grauwertbild (links), Binärbild (rechts) 106

12 TABELLENVERZEICHNIS

Tabelle 4-1: Auszug aus der Datei „Setup.txt“ 19

Tabelle 4-2: Auszug aus der Datei „Results.txt“ 20

Tabelle 4-3: Abweichungen der einzelnen Messungen 28

Tabelle 6-1: Datenblatt Intel Realsense D435 40

Tabelle 7-1: Messergebnis Aussetzer in der Pflanzenreihe 15 ± 2 cm Pflanzenabstand 45

Tabelle 7-2: Messergebnis Aussetzer der Pflanzenreihe 50 ± 2 cm Pflanzenabstand und variierende Verteilung 47

Tabelle 7-3: Messergebnis Variation der Pflanzen 49

Tabelle 7-4: Messergebnis Variation der Unkrautdichte 50

Tabelle 7-5: Messergebnis Linsenverschmutzung 59

Tabelle 7-6: Auflistung der durchgeführten Messreihen 62

Tabelle 7-7: Vergleich der Messergebnisse mit Pflanze 1 (P1) und Pflanze 2 (P2) und jeweils Boden 2 63

Tabelle 7-8: Vergleich der Messergebnisse mit Boden 1 (S1) und Boden 2 (S2) und jeweils Pflanze 2 64

13 LITERATURVERZEICHNIS

[1] A. Stentz, C. Dima, C. Wellington, H. Herman, and D. Stager, “A system for semi-autonomous tractor operations,” *Auton. Robots*, vol. 13, no. 1, pp. 87–104, 2002, doi: 10.1023/A:1015634322857.

[2] D. C. Slaughter, D. K. Giles, and D. Downey, “Autonomous robotic weed control systems: A review,” *Comput. Electron. Agric.*, vol. 61, no. 1, pp. 63–78, 2008, doi: 10.1016/j.compag.2007.05.008.

[3] H. Auernhammer, “Precision farming - The environmental challenge,” *Comput. Electron. Agric.*, vol. 30, no. 1–3, pp. 31–43, 2001, doi: 10.1016/S0168-1699(00)00153-8.

[4] M. Gärtner, M. Asbeck, S. Drüppel, K. Skindelies, and Stein;Markus, *Fachbuch für Vermessungstechniker und Geomatiker*. Gärtner, Michael, 2012.

[5] L. S. Guo and Q. Zhang, “A low-cost integrated positioning system for autonomous off-highway vehicles,” *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 222, no. 11, pp. 1813–1825, 2008, doi: 10.1243/09544070JAUTO250.

[6] S. Han, Q. Zhang, and H. Noh, "Kalman filtering of DGPS positions for a parallel tracking application," *Trans. Am. Soc. Agric. Eng.*, vol. 45, no. 3, pp. 553–559, 2002, doi: 10.13031/2013.8856.

[7] K. Neumann-Cosel, E. Roth, D. Lehmann, J. Speth, and A. Knoll, "Testing of image processing algorithms on synthetic data," *4th Int. Conf. Softw. Eng. Adv. ICSEA 2009, Incl. SEDES 2009 Simp. para Estud. Doutor. em Eng. Softw.*, no. July, pp. 169–172, 2009, doi: 10.1109/ICSEA.2009.34.

[8] S. Demers, P. Gopalakrishnan, and L. Kant, "A generic solution to software-in-the-loop," *Proc. - IEEE Mil. Commun. Conf. MILCOM*, 2007, doi: 10.1109/MILCOM.2007.4455268.

[9] R. Deutsch, "Vision- Based Crop Row Detection and Obstacle Recognition for Precision Farming," 2019.

[10] T. J. Monaco, A. S. Grayson, and D. C. Sanders, "Influence of Four Weed Species on the Growth, Yield, and Quality of Direct-Seeded Tomatoes (Lycopersicon esculentum)," *Weed Sci.*, vol. 29, no. 4, pp. 394–397, Jul. 1981, doi: 10.1017/s0043174500039874.

[11] J. M. Chandler and F. T. Cooke, "Economic of Cotton losses Cause by Weeds," in *Weeds of Cotton: Characterization and Control*, C. G. McWhorter and J. R. Abernathy, Eds. The Cotton Foundation, 1992, pp. 85–116.

[12] G. Elkaim, M. O'Connor, T. Bell, and B. Parkinson, "System identification and robust control of farm vehicles using CDGPS," *Proc. ION GPS*, vol. 2, pp. 1415–1424, 1997.

[13] M. Kise and Q. Zhang, "Development of a stereovision sensing system for 3D crop row structure mapping and tractor guidance," *Biosyst. Eng.*, vol. 101, no. 2, pp. 191–198, 2008, doi: 10.1016/j.biosystemseng.2008.08.001.

[14] J. A. Marchant, "Tracking of row structure in three crops using image analysis," *Comput. Electron. Agric.*, vol. 15, no. 2, pp. 161–179, 1996, doi: 10.1016/0168-1699(96)00014-2.

[15] M. Di Cicco, C. Potena, G. Grisetti, and A. Pretto, "Automatic Model Based Dataset Generation for Fast and Accurate Crop and Weeds Detection," no. September 2017, 2016.

[16] EpicGamesInc., "Unreal Engine." https://www.unrealengine.com/en-US/ (accessed Jul. 27, 2020).

[17] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "Fast robust monocular depth estimation for Obstacle Detection with fully convolutional networks," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4296–4303, 2016, doi: 10.1109/IROS.2016.7759632.

[18] M. Tschentscher, B. Prus, and D. Horn, "A simulated car-park environment for the evaluation of video-based on-site parking guidance systems," *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 1571–1576, 2017, doi: 10.1109/IVS.2017.7995933.

[19] T. Theuerkauff, T. Werner, F. Wallhoff, and T. Brinkhoff, "3D-Visualisierung von Über- und Unterwasserfahrzeugen zur Evaluation von Steuerungsalgorithmen mithilfe einer Game-Engine," no. September, 2017.

[20] Y. Zhang, W. Qiu, Q. Chen, X. Hu, and A. Yuille, "UnrealStereo: Controlling hazardous factors to analyze stereo vision," *Proc. - 2018 Int. Conf. 3D Vision, 3DV 2018*, pp. 228–237, 2018, doi: 10.1109/3DV.2018.00035.

[21] H. Behfar, H. Ghasemzadeh, A. Rostami, M. Seyedarabi, and M. Moghaddam, "Vision-Based Row Detection Algorithms Evaluation for Weeding Cultivator Guidance in Lentil," *Mod. Appl. Sci.*, vol. 8, no. 5, p. 224, 2014, doi: 10.5539/mas.v8n5p224.

[22] M. Haselhoff and S. Hakuli, "ECUs für kamerabasierte Fahrerassistenzsysteme im Closed-Loop-Verfahren," *ATZextra*, vol. 20, no. S7, pp. 30–33, 2015, doi: 10.1007/s35778-015-0002-4.

[23] J. Gemerek, S. Ferrari, B. H. Wang, and M. E. Campbell, "Video-guided Camera Control for Target Tracking and Following," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 176–183, 2019, doi: 10.1016/j.ifacol.2019.01.062.

[24] J. Meyer, J. Schüling, and A. Giersiefer, "Neue Methoden zur Entwicklung von Assistenzsystemen," *ATZextra*, vol. 17, no. 5, pp. 100–103, 2012, doi: 10.1365/s35778-012-0723-6.

[25] P. Smith, D. B. Reid, C. Environment, L. Palo, P. Alto, and P. L. Smith, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man. Cybern.*, vol. C, no. 1, pp. 62–66, 1979, doi: 10.1109/TSMC.1979.4310076.

[26] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, 2015.

[27] "Unreal_vision_bridge," 2018. https://github.com/sherpa-eu/unreal_vision_bridge (accessed May 08, 2020).

[28] J. Richartz, *Spiele entwickeln mit Unreal Engine 4*. München: Carl Hanser Verlag GmbH & Co. KG, 2016.

[29] V. Fontaine and T. G. Crowe, "Development of line-detection algorithms for local positioning in densely seeded crops," *Can. Biosyst. Eng. / Le Genie des Biosyst. au Canada*, vol. 48, pp. 19–29, 2006.

[30] L. Fahrmeir, T. Kneib, and S. Lang, *Regression, Modelle Methoden und Anwendungen*. .

[31] M. Neumayer, G. Steiner, and T. Bretterklieber, "Vorlesungsunterlagen Prozessinstrumentierung," 2017.

[32] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag New York, 1978.

[33] EpicGamesInc., "Sun and Sky Actor," *Unreal Engine Documentation*. https://docs.unrealengine.com/ (accessed May 05, 2020).

[34] "laengengrad-breitengrad.de." https://www.laengengrad-breitengrad.de/ (accessed May 05, 2020).

[35] "Roscore," *ROS Documentation*. http://wiki.ros.org/roscore (accessed Jul. 27, 2020).

[36] T. Wiedemeyer, "URoboVision," *UnrealCV Project*. https://github.com/robcog-iai/URoboVision (accessed May 17, 2020).

[37] ROS Community, "ROS Bridge," *wiki.ros.org*. http://wiki.ros.org/rosbridge_suite (accessed Apr. 30, 2020).

[38] "bsonspec." http://bsonspec.org/ (accessed Apr. 07, 2020).

[39] EpicGamesInc., "Sky Light," *Unreal Engine Documentation*.

https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightTypes/SkyLight/index.html (accessed May 06, 2020).

[40] EpicGamesInc., "Directional Lights," *Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightTypes/Directional/index.html (accessed May 05, 2020).

[41] EpicGamesInc., "Sky Atmosphere," *Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Engine/Actors/FogEffects/SkyAtmosphere/index.html (accessed May 05, 2020).

[42] EpicGamesInc., "Exponential Height Fog User Guide," *Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Engine/Actors/FogEffects/HeightFog/index.html?utm_source=editor&utm_medium=docs&utm_campaign=doc_anchors (accessed May 05, 2020).

[43] EpicGamesInc., "Using Cameras," *Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Gameplay/HowTo/UsingCameras/index.html (accessed Jun. 17, 2020).

[44] OmniVision, "Image Sensor - OV2740," 2019, [Online]. Available: https://www.ovt.com/download/sensorpdf/153/OmniVision_OV2740.pdf.

[45] O. Schreer, *Stereoanalyse und Bildsynthese*. Berlin/Heidelberg: Springer-Verlag, 2005.

[46] B. Åstrand and A. J. Baerveldt, "A vision based row-following system for agricultural field machinery," *Mechatronics*, vol. 15, no. 2, pp. 251–269, 2005, doi: 10.1016/j.mechatronics.2004.05.005.

[47] H. Tümmel, *Die Wissenschaftliche und Angewandte Photographie: Erneuerung und Fortführung des Hay- v. Rohrschen Handbuchs der Wissenschaftlichen und Angewandten Photographie Sechster Band Laufbildprojektion*. Springer Vienna, 1973.

[48] A. Amer and H. Schroeder, "New video noise reduction algorithm using spatial subbands," *Proc. IEEE Int. Conf. Electron. Circuits, Syst.*, vol. 1, pp. 45–48, 1996, doi: 10.1109/icecs.1996.582658.

[49] A. Amer and E. Dubois, "Fast and reliable structure-oriented video noise estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 113–118, 2005, doi: 10.1109/TCSVT.2004.837017.

[50] A. Erhardt, *Einführung in die Digitale Bildverarbeitung - Grundlagen, Systeme und Anwendungen*, vol. 1, no. c. 2015.

[51] Intel, "Intel® RealSense™ Camera D400 series Product Family Datasheet Rev. 01/2019," 2019, [Online]. Available: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf.

[52] K. Schauwecker, "Real-Time Stereo Vision on FPGAs with SceneScan," pp. 1–12, 2018, [Online]. Available: http://arxiv.org/abs/1809.07977.

[53] S. Su and W. Heidrich, "Rolling shutter motion deblurring," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1529–1537, 2015, doi: 10.1109/CVPR.2015.7298760.

[54] Intel Corportation, "Intel RealSense Depth Camera D400-Series," no. September, 2017.

[55] M. Bahadir, B. B. Forschung, H. Parlar, and M. Spiteller, *Springer Umweltlexikon*. Springer Berlin Heidelberg, 2013.

[56] EpicGamesInc., "Bloom," *Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Engine/Rendering/PostProcessEffects/Bloom/index.html (accessed May 06, 2020).

[57] P. Seitz, "Solid-State Image Sensing," in *Computer Vision and Applications*, 1st ed., B. Jähne, H. Haußecker, and P. Geißler, Eds. 1999, pp. 165–222.

[58] "Universität Hohenheim." https://www.uni-hohenheim.de/portraet (accessed Jun. 03, 2020).

[59] D. Riehle, D. Reiser, and H. W. Griepentrog, "Robust index-based semantic plant/background segmentation for RGB- images," *Comput. Electron. Agric.*, vol. 169, no. January, p. 105201, 2020, doi: 10.1016/j.compag.2019.105201.

[60] S. by Adobe, "Substance products." https://www.substance3d.com/products/substance-source/ (accessed Jun. 15, 2020).

[61] ArtStation, "Artstation farm material." https://www.artstation.com/emrecancubukcu/store/8LRd/farm-material-100-substance-designer (accessed Apr. 07, 2020).

[62] K. Bikos, "Finsternisse: Was ist der Halbschatten." https://www.timeanddate.de/finsternis/penumbra-halbschatten (accessed Apr. 30, 2020).

[63] A. Erhardt, *Einführung in die Digitale Bildverarbeitung*. Wiesbaden: Vieweg+Teubner, 2008.

[64] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn, "Differences between stereo and motion behaviour on synthetic and real-world stereo sequences," *2008 23rd Int. Conf. Image Vis. Comput. New Zealand, IVCNZ*, no. June 2014, 2008, doi: 10.1109/IVCNZ.2008.4762133.

[65] EpicGamesInc., "Auto Exposure (Eye Adaptation)." https://docs.unrealengine.com/en-US/Engine/Rendering/PostProcessEffects/AutomaticExposure/index.html (accessed May 14, 2020).