Fachbereich 3 - Mathematik und Informatik

Bestimmung der 3D-Pose eines fliegenden Kites basierend auf 2D-Luftaufnahmen für Airborne Wind Energy Anwendungen

Bachelorarbeit

im Studiengang Systems Engineering
Spezialisierung Robotik und Automatisierungstechnik

Autor: Nils Öhlmann <nils.oehlmann@uni-bremen.de>
Matr.-Nr. 2677872

Datum: 21. August 2015

1. Gutachter: Prof. Dr. U. Frese
2. Gutachter: Priv. Doz. Dr.-Ing. habil. B. Gottfried

Eidesstattliche Erklärung zur Bachelorarbeit

Ich versichere durch meine Unterschrift, dass ich die Arbeit selbstständig verfasst habe und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Die Arbeit oder Auszüge daraus haben noch nicht in gleicher oder ähnlicher Form dieser oder einer anderen Prüfungsbehörde vorgelegen.

Ich weiß, dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat.

Bremen, 21. August 2015

Unterschrift

Abstract

Bestimmung der 3D-Pose eines fliegenden Kites basierend auf 2D-Luftaufnahmen für Airborne Wind Energy Anwendungen

Diese Bachelorarbeit beschäftigt sich mit der Bestimmung der Pose (Orientierung und Position) eines fliegenden Kites mit Mitteln der Bildverarbeitung. Dazu werden Videos mit einer am Kite montierten GoPro Hero-Kamera ausgewertet, die den Boden um die Bodenstation des Kites aufnehmen. Die Berechnung basiert dabei auf am Boden platzierten, quadratischen Schachbrettmarkern, um die Pose mithilfe der Lösung des Perspective-n-Point-Problems zu ermitteln. Die berechnete Pose soll in Zukunft als Verifikation für vorhandene Messwerte einer Inertial Measurement Unit zur Posenberechnung eines Prototypen im Bereich der Airborne Wind Energy dienen. Dazu wurde unter Verwendung der Bildverarbeitungsbibliothek OpenCV in C++ ein Programm entwickelt, das es für Videos des Kiteflugs ermöglicht, die Pose für die einzelnen Frames zu berechnen. Anhand einer ebenfalls angefertigten Simulation der Flugphase des Kites und anhand von Modelltests mit einer realen Kamera im kleinen Maßstab konnte damit die Pose des Kites erfolgreich ausreichend genau bestimmt werden.

Estimation of the 3D Pose of a Tethered Kite based on 2D Aerial Images for Airborne Wind Energy Applications

This thesis aims to develop an approach for estimating the pose (orientation and position) of a tethered kite using image processing methods. This is done by analyzing video data recorded with a GoPro Hero camera that is mounted to the kite in flight. The calculation is based on square chessboard markers that are located on the ground area around the kite's ground station. The calculated pose is to be used in the future to verify existing pose data generated by an Inertial Measurement Unit during the autonomous flight of a tethered kite prototype for Airborne Wind Energy. For that purpose, a C++ application was developed that utilizes the computer vision library OpenCV and is capable of calculating the pose for each frame of the video of the kite's flight. Using a simulation of the kite's flight and a small scale model, which were also developed as a part of this thesis, the pose was successfully calculated with sufficient accuracy for the required task.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AR	Augmented Reality
ASCII	American Standard Code for Information Interchange
AWE	Airborne Wind Energy
CSV	Comma Separated Values
DGPS	Differential Global Positioning System
FOV	Field Of View
FPS	Frames per second
GPS	Global Positioning System
IMU	Inertial Measurement Unit
kWh	Kilowattstunde
MEMS	Microelectromechanical Systems
MW	Megawatt
PnP	Perspective-n-Point
QR-Code	Quick Response Code
RANSAC	Random Sample Consensus
SfM	Structure from Motion
UAV	Unmanned Aerial Vehicle, z. Dt. unbemanntes Luftfahrzeug
WEA	Windenergieanlage

1 Einleitung

Eine der wichtigsten erneuerbaren Energien ist die Windenergie, die in Deutschland zunehmend an Bedeutung gewinnt. Sie ist 2014 für 8,6 % bzw. 52,4 Mrd. kWh des Strombedarfs der Bundesrepublik verantwortlich und damit führend unter den erneuerbaren Energien ([Age15]). Dabei wird die Windenergie zumeist in Windenergieanlagen (WEA) gewonnen, die heute vielerorts das Landschaftsbild in Deutschland prägen. Allerdings können sie für Anwohner unter anderem eine Lärmbelästigung darstellen und sind nicht zuletzt wegen ihrer enormen Masse teuer in Bau und Unterhaltung.

Das sind einige Gründe, warum sich insbesondere seit der Jahrtausendwende neben den herkömmlichen WEA ein weiteres Feld im Bereich der Windenergie stark weiterentwickelt hat: die Airborne Wind Energy (AWE). Unter dem Begriff der Airborne Wind Energy werden verschiedene Systeme zusammengefasst, die sich im Unterschied zu herkömmlichen WEA entweder freifliegend in der Luft bewegen oder über ein Seil mit dem Boden verbunden sind. Dabei haben sie zum Ziel, Winde auch in größeren Höhen auszunutzen und den Materialbedarf deutlich zu senken.

Moderne WEA verfügen heutzutage über eine Nennleistung von 7,5 MW (Enercon E-126). Um diese Leistung erreichen zu können, wiegt ein einzelnes Rotorblatt jedoch bereits 65 t. Interessant ist dabei, dass die äußeren, spitz zulaufenden, 30 % der Rotorblätter über die Hälfte der Energie erzeugen (vgl. [Die13], S. 5). Die kleineren und zugleich leichteren Teilflächen der Rotorblätter liefern also einen großen Teil der Energiegewinnung. Diese Erkenntnis stellt die Grundidee von AWE-Systemen dar: Anstatt schwere, in der Anschaffung teure, Komponenten zur Energiegewinnung zu nutzen, werden lediglich die sich am schnellsten bewegenden Teile, die Spitzen der Rotorblätter, durch automatisiert fliegende Kites oder Tragflügel ersetzt. Verallgemeinert ausgedrückt sollen der Turm sowie die inneren Teile der Rotorblätter durch ein mit dem Boden verbundene Seil ersetzt werden (vgl. [Die13], S. 5).

Dabei liegt die auf Simulationen basierende theoretische maximale Leistung eines AWE-Systems bei 40 kW/m^2 Flügelfläche ([Die13], S. 5). Wenngleich dieser Wert in der Praxis bisher nicht erreicht werden konnte, ermöglicht er theoretisch eine über 150-fach höhere Leistung als moderne Photovoltaikanlagen bei einer Masse, die etwa 0,3 % der einer WEA entspricht ([Die13], S. 6). Aufgrund ihrer leichten Bauweise könnten sie in Zukunft auch für Offshore-Anwendungen in größeren Meerestiefen interessant werden: ein schwimmender Ponton, der über Seile am Meeresboden verankert wird, wäre ausreichend, um das System zu befesteigen. Dadurch würde eine aufwendige Befestigung des Fundaments am Meeresboden, wie es heutzutage in flacheren Gewässern für WEA üblich ist, entfallen.

AWE-Systeme werden in eine Reihe von Kategorien gegliedert, die sich hauptsächlich nach Art der Energiegewinnung unterscheiden. Neben anderen Verfahren wird bei der On-Board-Stromerzeugung beispielsweise direkt am Kite bzw. an einem kabelgebundenen Flugzeug Energie über eine auf dem Fluggerät montierte Windturbine gewonnen. Dies impliziert jedoch eine permanente Verbindung zum Boden, die neben der Eingrenzung der Bewegung des Kites auch für den Transfer der gewonnenen elektrischen Energie verantwortlich ist, und stellt eine große Herausforderung an das Material des verwendeten Seils dar.

Demgegenüber stehen Systeme, die auf eine Energiegewinnung am Boden setzen. Eine Möglichkeit bietet hier ein Generator am Boden, der mit dem Seil eines fliegenden Kites verbunden ist. Während sich der Kite autonom in definierten Flugbahnen bewegt, wird das Seil von einer Winde abgewickelt und treibt so den Generator an. Da das Seil auf eine endliche Länge beschränkt ist, muss bei dieser Art der Nutzung das Seil in regelmäßigen Abständen eingeholt werden. Dieser kontinuierliche Kreislauf zur Energiegewinnung wird auch als Pumping Cycle bezeichnet und setzt sich aus der Reel-Out- oder Energieerzeugungsphase zur Energiegewinnung, der Transferphase, in der der Kite in eine Position geflogen wird, in der eine energieeffiziente Rückholung des Kites möglich ist, sowie der Reel-In- oder Rückholphase, in der der Kite schließlich eingeholt wird, zusammen (vgl. [Die13], S. 9).

Anwendung findet dieses System in Deutschland bei der SkySails GmbH & Co. KG, die seit ihrer Gründung 2001 im AWE-Bereich aktiv ist. Dabei wurde zunächst kein stationärer Generator zur Gewinnung elektrischer Energie genutzt. Vielmehr wurde das SkySails Marine genannte System mit dem Ziel entwickelt, einen Zugdrachen mit Seil zu nutzen, um den Wind als Antriebsquelle für Frachtschiffe nutzbar zu machen und so die Treibstoffkosten sowie Emmissionen zu senken.

Diese Arbeit ist konkret auf einen Prototypen in Form eines kabelgebundenen Kites von SkySails zur stationären Energiegewinnung ausgelegt, der bei einer Größe von 30 m^2 und einer Seillänge von 130 – 300 m eine 50 kW Motor- und Generatorkombination antreibt (Abb. 1).

Unabhängig vom eingesetzten AWE-System ist ein entscheidender Aspekt die komplexe Regelung der autonomen Flugphase, die auf plötzlich auftretende Umwelteinflüsse immer zuverlässig und vor allem schnell reagieren muss, um einen sicheren Betrieb zu gewährleisten. Bei einem Vergleich von WEA und AWE-Systemen wird dabei ein stabiles System im Fall der WEA, das bei Bedarf sicher unterbrochen werden kann, durch ein instabiles AWE-System ersetzt, das bei Fehlfunktionen nicht ohne erhöhten Aufwand sicher gestoppt werden kann (vgl. [Die13], S. 6).

Aus diesem Grund ist eine ausgereifte Regelung unerlässlich für AWE-Systeme. Der konkret von SkySails genutzte Ansatz zur Regelung besteht aus einem Kontrollsockel

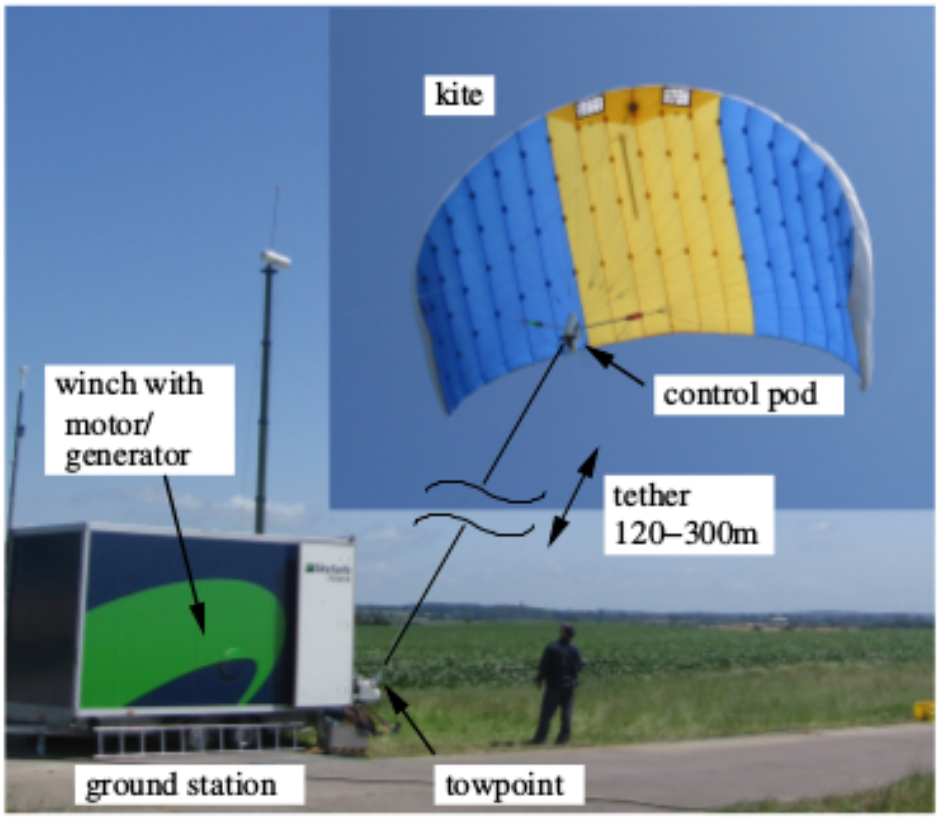


Abbildung 1: SkySails-Prototyp, dessen Pose im Rahmen dieser Arbeit bestimmt wird ([ES15], S. 2)

(Control Pod) unterhalb des Kites, in dem die zwei Seilstücke zur Steuerung des Kites zusammenlaufen. Vergleichbar mit einem herkömmlichen Freizeitlenkdrachen kann über die Variation der jeweiligen Seillänge das Flugverhalten des Kites direkt beeinflusst werden. Der offensichtliche Vorteil des Kontrollsockels direkt unterhalb des Kites liegt darin, dass unterhalb des Sockels ein einzelnes Seil als Verbindung zum Boden ausreicht und nicht länger zwei Seile zur Kontrolle vom Boden benötigt werden.

Die Regelung ist dabei abhängig von der Rückführung aktueller Messwerte zum Istzustand des Kites, um auf aktuelle Abweichungen und Störeinflüsse reagieren zu können. Eine entscheidende Messgröße stellt insbesondere die aktuelle Position und Orientierung (Pose) des Kites dar, da diese direkt die Bewegung beeinflusst. Insbesondere die Bestimmung der Orientierung des Kites gilt dabei als wichtigste Aufgabe in der Entwicklung einer Regelung für AWE-Systeme ([ES13], S. 998). Aus diesem Grund sollen die vom Protoypen verwendeten Messverfahren zur Ermittlung der Pose (siehe Kapitel 2) durch ein alternatives Verfahren verifiziert werden.

Ziel dieser Bachelorarbeit ist es somit, die Frage zu beantworten, ob es möglich ist, mit vertretbarem Aufwand die 3D-Pose eines Kites mit einer Seillänge von 130 – 300 m mit einer am Kite montierten Kamera ausreichend genau zu bestimmen. Dazu ist die Arbeit als Vorstudie ausgelegt, in der ein Programm zur Posenbestimmung basierend

auf am Boden platzierten Schachbrettmarkern entwickelt wurde, mit dem es in ersten Tests möglich war, die Pose in Höhen des Prototypen auf unter 5 m in der Position bzw. unter 1° in der Orientierung genau zu bestimmen (siehe Kapitel 7). Anschließend erfolgt eine Evaluation der Messergebnisse des Programms anhand einer Simulation der Flugphase (Pumping Cycle) des Kites sowie anhand eines Modells im Maßstab 1:100 anhand eines Kamerastativs, die beide ebenfalls im Rahmen dieser Arbeit umgesetzt wurden.

Die Arbeit ist so aufgebaut, dass in Kapitel 2 zunächst anhand einer Auswahl an Verfahren und Anwendungsbeispielen zur Posenberechnung der Stand der Technik erläutert wird, bevor in Kapitel 3 der gewählte Lösungsansatz vorgestellt wird. Kapitel 4 erläutert theoretisch das zur Umsetzung nötige Vorgehen und zugehörige Algorithmen, bevor in Kapitel 5 und 6 zunächst die Umsetzung des Programms und anschließend die der Simulation erläutert wird. Abschließend erfolgt die ausführliche Evaluation des entwickelten Programms sowie das Fazit in den Kapiteln 7 und 8.

2 Stand der Technik

Das Problem der Bestimmung von Position und Orientierung (Pose) eines Objekts relativ zu dessen Umgebung ist weit verbreitet und tritt in einer Vielzahl von verschiedenen Anwendungsfällen auf. Dabei unterscheiden sich nicht nur die Anwendungsfälle stark voneinander, sondern auch die jeweils angewandten Ansätze und Methoden zur Berechnung der Pose.

Viele Anwendungen setzen auf eine Kombination von Messsystemen zur getrennten Bestimmung von Position und Orientierung. Hier haben sich für die Positionsbestimmung Systeme aus der Satellitennavigation, allen voran dem amerikanischen NAVSTAR Global Positioning System (häufig gekürzt GPS), bewährt. Häufig wird dieses System in Kombination mit einer Interial Measurement Unit (IMU) genutzt, die zur Bestimmung der Orientierung geeignet ist.

GPS und IMUs werden dabei heutzutage beispielsweise zur Steuerung von unbemannten Luftfahrzeugen (engl. Unmanned Aerial Vehicle, UAV) und in der Navigation eingesetzt. Dazu zählen Anwendungen im Bereich der allgemeinen Luftfahrt, Raumflugkörper oder Schiffe. Aber auch in modernen Smartphones und anderen tragbaren Geräten kommen IMUs zum Einsatz, um die Orientierung des Geräts in seiner Umgebung zu erfassen und entsprechend zu reagieren. Ein Beispiel ist hier die automatische Rotation des Bildschirminhalts, wenn der Nutzer das Gerät entsprechend dreht. Ein weiteres konkretes Anwendungsbeispiel ist die Umsetzung einer zusätzlichen Einheit für streifenprojektionsbasierte, portable 3D-Scanner, die es ermöglicht, die aufgenommenen Teilansichten automatisch zusammenzufügen (vgl. [MKNK14]). Das automatische Zusammenfügen basiert dabei auf den ermittelten Posendaten der IMU der Zusatzeinheit.

Das weit verbreitete GPS-System zur Positionsbestimmung und Geschwindigkeitsmessung ermittelt die Position eines Objekts dabei über Entfernungsmessungen zu einer definierten Anzahl an Satelliten. Um die Entfernung eines einzelnen Satelliten zur GPS-Empfangsantenne am Objekt zu berechnen, wird die Zeitdifferenz vom Aussenden des Signals am Satelliten bis hin zum Empfang des Signals durch die Empfangsantenne genutzt. Dazu sind die Satelliten mit Atomuhren ausgestattet, die eine präzise Zeiterfassung ermöglichen. Drei Punkte bzw. Satelliten sind im Normalfall ausreichend, um eine eindeutige Position auf der Oberfläche der Erde zu bestimmen. Allerdings verfügen die Empfänger nicht über Atomuhren und können somit keine absolute Zeitdifferenz berechnen. Daher wird eine Zeitkonstante als weitere Unbekannte neben den drei Positionskoordinaten eingeführt. Allgemein werden aus diesem Grund vier Satelliten zur Positionsbestimmung mittels GPS benötigt.

Aktuell liegt die Genauigkeit im zivilen Bereich bei unter 10 m. Je nach Anwendung ist dies nicht immer ausreichend genau. Um die Genauigkeit zu erhöhen, kann ein Differential Global Positioning System (DGPS) genutzt werden. In diesem Fall

werden Korrekturdaten über eine lokale Referenzstation erzeugt, um die allgemeine GPS-Position zu verbessern. Qualitativ hochwertige Systeme erreichen so Genauigkeiten im Zentimeterbereich.

Ein möglicher Nachteil sind jedoch mögliche Störungen des GPS-Signals durch eine nicht immer gegebene Satellitenabdeckung insbesondere in urbanen Gegenden in relativer Nähe zur Erdoberfläche und Störeinflüsse insbesondere in dicht besiedelten Städten. Außerdem lassen sich auf Basis der Entfernungsmessungen wie bereits angedeutet lediglich Aussagen über die Position eines Objekts, nicht aber über dessen Orientierung machen.

Abhilfe schafft der Einsatz einer IMU zusätzlich zur GPS-Technik. IMUs sind Messsysteme zur Bestimmung der linearen Beschleunigungen sowie der Winkelgeschwindigkeiten bzw. Rotationswinkel im dreidimensionalen Raum. Dazu werden in der Regel jeweils drei orthogonal angeordnete Beschleunigungssensoren sowie Drehratensensoren verwendet.

Während frühere Geräte ein Gewicht von mehreren Kilogramm aufwiesen, sind heute leichte Bauweisen auf Basis von sogenannten Microelectromechanical Systems-Technologie (MEMS) möglich, die aus Bausteinen in Größenordnungen zwischen 1 und 100 μm bestehen. MEMS-Sensoren sind in der Anschaffung zwar zumeist kostengünstiger. Dies geht jedoch in einigen Fällen einher mit einer höheren Fehleranfälligkeit bzw. ungenaueren Messungen, als es bei herkömmlichen mechanischen oder optischen Drehraten- bzw. Beschleunigungssensoren der Fall wäre (vgl. [Woo07], S. 9). Aufgrund der rasanten Entwicklung der MEMS-Systeme werden aber auch diese Einschränkungen immer weiter aufgehoben. Insbesondere die deutlich leichtere und wesentlich kleinere Bauweise ist dabei relevant für AWE-Systeme, die generell auf eine möglichst geringe Masse und kompakte Größen ausgelegt werden.

Sowohl MEMS-Beschleunigungssensoren als auch -drehratensensoren sind anfällig gegenüber verschiedenen Störquellen. Dazu zählen u. a. weißes Rauschen, eine konstante Messabweichung sowie Temperatureinflüsse (vgl. [Woo07], S. 11 ff., 15 ff.). Dabei ist ein entscheidender Nachteil, dass diese Fehler bei der Nutzung eines IMU über die Zeit akkumuliert werden. Da beispielsweise zur Bestimmung der Orientierung die gemessenen Winkelgeschwindigkeiten über die Zeit integriert werden, hängen die aktuellen Messwerte von den vorherigen Messwerten mit ihren jeweiligen Messfehlern ab. Dieser sich akkumulierende Fehler wird als Drift bezeichnet und nimmt mit zunehmender Nutzungsdauer immer weiter zu.

Auch der von SkySails verwendete Prototyp, auf den diese Arbeit ausgelegt ist, nutzt ein IMU mit sechs Freiheitsgeraden (drei Drehratensensoren sowie drei Beschleunigungssensoren) zur Bestimmung der Orientierung des Kites. Zusätzlich wird ein 2D-Anemometer auf Ultraschallbasis in 5 m Höhe eingesetzt, das als Referenz für die IMU

am Kontrollsockel des Kites dient (vgl. [ES15], S. 2). Um den oben erwähnten Drift zu kompensieren, kommt in diesem Fall ein Complementary Filter zum Einsatz, bei dem zur Bestimmung der Orientierung die Daten von Drehratensensoren und Beschleunigungssensoren kombiniert genutzt werden.

Neben der sensorbasierten Bestimmung existiert eine Vielzahl von Ansätzen aus der Bildverarbeitung, die für verschiedene Anwendungsfälle eine kamerabasierte Posenberechnung ermöglichen. Dabei spielt der Bereich des maschinellen Sehens in der Robotik eine wesentliche Rolle. Dort reicht das Spektrum von Pick-and-Place-Aufgaben mittels Objekterkennung über Qualitätsprüfungen bis hin zur Kollisionsvermeidung, um nur ein paar Beispiele zu nennen. Außerdem spielt die Posenberechnung eine große Rolle in der Photogrammetrie. Ein Beispiel ist dabei der Prozess zur Berechnung von dreidimensionalen Strukturen basierend auf 2D-Bildsequenzen ggf. unter Verwendung lokaler Umgebungsinformationen, dem sogenannten Structure from Motion (SfM). Um Informationen über Strukturen im Bild zu erhalten, werden Merkmale wie beispielsweise Ecken eines Objekts bzw. einer Stuktur über eine Sequenz von Bildern extrahiert. Anschließend werden diese Informationen einzelner Merkmale aus verschiedenen Bildern bzw. Frames genutzt, um die Positionen der Merkmale sowie die Kamerabewegung zu berechnen. Entscheidend ist dabei die robuste Merkmalerkennung, für die eine Vielzahl von Feature Detectors mit verschiedenen Eigenschaften existiert. Dabei existieren Ansätze, die keine Marker zum Verfolgen von Objekten benötigen, sondern allgemeine Objektmerkmale im Bild erfassen und verfolgen. Dieser Ansatz ist jedoch für die Posenbestimmung des Kites ungeeignet, da zur Bestimmung der Pose auch die reale 3D-Position der Merkmale bekannt sein muss (siehe Kapitel 4.3). Ohne ausgelegte Marker am Boden wäre es nicht ohne Weiteres möglich, die reale Position von den im Bild erfassten Merkmalen zu bestimmen.

Insbesondere in den vergangenen Jahren hat außerdem der Bereich der erweiterten Realität (engl. Augmented Reality, AR) an Bedeutung gewonnen, in der virtuelle Objekte zumeist in Echtzeit in Aufnahmen der realen Umgebung eingebettet werden. Um die korrekte Ausrichtung der virtuellen Objekte zu erreichen, ist eine genaue Kenntnis der Pose unabdingbar. Häufig werden zu diesem Zweck planare Marker mit Quick Response (QR)-Codes genutzt, die Bitinformationen über das jeweils darzustellende Objekt enthalten. Auch hier ist das Tracken der Objektposition über mehrere Frames eines Videos nötig, um das Objekt als stationäres Objekt innerhalb der Aufnahme darzustellen. Ebene Marker können dabei über eine 2D-Homographie erkannt und ausgewertet werden, um anschließend virtuelle Objekte darzustellen. Bei dieser Art von Markern ist allerdings die Skalierung auf große Entfernungen problematisch, da die Marker ausreichend hoch aufgelöst sein müssen, um die einzelnen Bits des Markers zu erkennen. Daher ist dieser Ansatz für die Posenbestimmung des Kites ungeeignet, da

die Marker eine nur schwierig zu realisierende Größe aufweisen müssten, um aus bis zu 300 m korrekt erkannt werden zu können.

Wie bereits zuvor erwähnt hat die Bedeutung von UAVs besonders im 21. Jahrhundert stark zugenommen. Um Flugmanöver wie Landungen oder das Abladen einer Nutzlast präzise autonom durchführen zu können, spielt die Posenerkennung mittels Bildverarbeitung mittlerweile auch hier eine Rolle. Dabei ist die Posenbestimmung eines fliegenden UAVs in Teilen vergleichbar mit der Posenerkennung des Kites. Der Ansatz von Mondragon et al. ([MCMOM10]) basiert dabei auf der stückweisen Erkennung von ebenen Objekten über robuste Homographien. Als Referenzobjekt dient bei diesem Ansatz ein ebener Marker in Form eines Hubschrauberlandeplatzes. Dazu werden zunächst die Ecken des Landeplatzes mit dem Good Features to Track-Algorithmus erkannt und dann mit dem pyramidalen Lucas Kanade Optical Flow-Algorithmus die zuvor erkannten Eckpunkte verfolgt (vgl. [MCMOM10], S. 37 f). Dieser Algorithmus bietet eine robuste Merkmalerkennung auch bei schnellen Bewegungen. Beide Algorithmen werden auch in der Umsetzung dieser Arbeit verwendet. Anschließend wird daraus eine Homographie ermittelt, die durch die Verwendung des Random Sample Consensus (RANSAC)-Algorithmus robust bestimmt werden kann. RANSAC wird im Rahmen dieser Arbeit ebenfalls zur robusten Markerdetektion evaluiert (siehe Kapitel 4.2).

Allerdings ist dieser Ansatz wie viele weitere abhängig von externen Einflüssen wie Lichteinfall und Witterung und hängt insbesondere von der Sichtbarkeit des Referenzobjekts ab. Konkret geben Mondragon et al. an, dass mindestens 30 % ihres Landeplatzes zur korrekten Posenbestimmung sichtbar sein müssen (vgl. [MCMOM10], S. 41).

Abhilfe zur Abhängigkeit von Witterung und anderen Störfaktoren bietet ein Ansatz von Faessler et al. ([FMSS14]), bei dem stattdessen ein Zielobjekt (z. B. eine Drohne) mit LEDs im Infrarotbereich bestückt wird. Eine Kamera mit Infrarot-Passfilter, die beispielsweise auf einem mobilen Roboter montiert ist, erfasst die LEDs unter Verwendung eines simplen Schwellwerts. Dies erleichtert zum einen die Markerdetektion und bietet zum anderen Unabhängigkeit von Störeffekten, die sich auf eine herkömmliche Kamera auswirken würden. Dieser Ansatz soll vor allem für den Einsatz in Katastrophengebieten unter Verwendung von kollaborierenden Robotern sowohl in der Luft als auch am Boden genutzt werden, um schnell und zuverlässig Informationen der Lage liefern zu können. Faessler et al. beschränken ihre Versuche allerdings auf eine Höhe von 5 m (vgl. [FMSS14], S. 911), sodass große Höhen auch hier ein Problem darstellen könnten.

3 Lösungsansatz

Unter Verwendung von Methoden der Bildverarbeitung ist es mit relativ geringem materiellen Aufwand möglich, die Pose eines Objekts zu bestimmen. Im Rahmen dieser Arbeit wird dazu ein Lösungsansatz zur Berechnung der Pose gewählt, der auf 2D-Luftaufnahmen von punktförmigen Schachbrettmarkern basiert. Dazu werden die Marker (Abb. 3) in der Umgebung der Basisstation des Kites ausgelegt, um von der am Kite montierten Kamera während des Fluges erfasst zu werden. Abb. 2 zeigt eine Aufnahme der realen, am Kite montierten, Kamera, allerdings ohne am Boden platzierte Marker.

Abbildung 2: Reale Aufnahme der am Kite montierten Kamera ohne am Boden ausgelegte Marker

Über die Korrespondenz der realen Markerpositionen und der jeweils zugehörigen 2D-Position im aktuellen Bild ist es dann möglich, die Pose zu berechnen, sofern eine ausreichende Anzahl an Markern sichtbar ist (siehe Kapitel 4.3). Dabei ist es entscheidend, sowohl die reale Position als auch die Bildposition der einzelnen Marker so genau wie möglich zu bestimmen, da diese maßgeblich die Genauigkeit der Posenberechnung beeinflussen. Schachbrettmarker sind zur Bestimmung der genauen Bildposition gut geeignet, weil sie über eine ausgeprägte innere Kante verfügen, die über den extremen Gradientenverlauf an den Kantenübergängen eine robuste und präzise Bestimmung der Bildposition bis in den Subpixelbereich ermöglichen (siehe Kapitel 4.2). Außerdem sind sie in ihrer Größe theoretisch beliebig skalierbar und somit auch für größere Entfernungen geeignet. Ein weiterer positiver Aspekt bei der Verwendung der Schachbrettmarker ist zudem die kostengünstige Umsetzung beispielsweise in Form von Markern aus Stoff, die aus quadratischen Einzelflächen zusammengesetzt werden und sich leicht platzieren und lagern lassen. Die Marker werden dabei in der Simulation bzw. im Modell mit einer Größe von 2,8 m x 2,8 m bzw. 2,8 cm x 2,8 cm umgesetzt. Diese Größe wurde gewählt,

da der mögliche Stoff für die Marker jeweils mit einer Breite von 1,4 m zu erwerben ist.

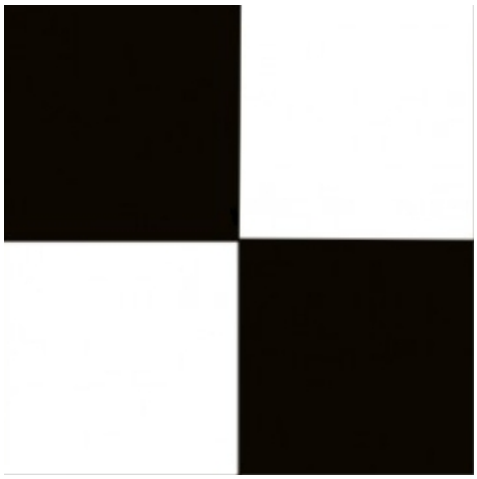


Abbildung 3: Idealer Marker in der Simulation

Das Programm zur Bestimmung der Pose gliedert sich also in die Hauptaufgaben zunächst der Bestimmung der Markerpositionen im jeweiligen Bild und der anschließenden Berechnung der Pose unter Verwendung der 2D-3D-Punktkorrespondenzen der einzelnen Marker. Dazu soll es mit dem im Rahmen der Arbeit entwickelten Programm anhand von Videosequenzen von ca. 10 Minuten Länge effizient möglich sein, die Pose des Kites für die Einzelbilder (Frames) des Videos bzw. davon abgeleitet zu einem bestimmten Zeitpunkt innerhalb des Videos zu bestimmen. Bei dem Bildmaterial handelt es sich dabei nicht um Livebilder. Stattdessen sollen die Videos im Anschluss an den Kiteflug als Verifizierung bisheriger Posendaten ausgewertet werden. Für diese Auswertung stehen verschiedene Lösungsansätze zur Auswahl, die das gewünschte Ergebnis erzielen können.

Ein solcher, in der Umsetzung simpler Ansatz ist die manuelle Posenberechnung für einzelne Frames. In diesem Fall wählt der Nutzer selbstständig die inneren Kanten der Marker beispielsweise per Mausklick im Bild aus und startet anschließend die Berechnung für den aktuellen Frame. Vorteilhaft ist dabei sicherlich die mögliche Fehlervermeidung, die eine Erkennung der Marker ausgehend vom Nutzer im Vergleich zu Algorithmen zur Kantendetektion bietet. Bei Letzteren sind fälschlicherweise als Marker erkannte Bildmerkmale nicht auszuschließen. Allerdings geht dies einher mit einem enormen Arbeitsaufwand auch bei kurzen Videosequenzen: bei einem Video von zehn Minuten Länge und einer Bildfrequenz von 30 FPS müssten 18000 Einzelbilder manuell bearbeitet werden. In der Praxis ist dieser Ansatz daher allenfalls dann brauchbar, wenn keine kontinuierlichen Posendaten ermittelt werden sollen.

Demgegenüber steht die Möglichkeit, die Markerauswahl und anschließende Berechnung automatisiert über alle Frames zu implementieren. Dieser Ansatz stellt die für

den Benutzer komfortabelste Lösung dar. Allerdings stehen dem mehrere Schwierigkeiten in der Umsetzung gegenüber. Zunächst ist es zwingend erforderlich, die Marker im Bild korrekt zu erkennen und über alle Frames zu verfolgen, da ansonsten fehlerhafte Posenberechnungen die Folge wären, die der Nutzer erst nach Abschluss der Berechnung aller Frames erkennen würde. Während diese Herausforderung sehr schwierig, aber nicht unmöglich zu bewältigen ist, sorgt ein weiteres Problem dafür, dass ein vollständig automatisierter Ansatz für den konkreten Anwendungsfall nicht in Frage kommt: Die Posenberechnung basierend auf einheitlichen Schachbrettmarkern am Boden benötigt die Zuordnung der realen 3D-Position eines Markers zu dessen aktueller Position im Bild (siehe Kapitel 4.3). Ohne Weiteres sind die erkannten Marker im Bild jedoch nicht zu unterscheiden und können dementsprechend nicht automatisiert ihrer Position im Bild zugeordnet werden. Abhilfe könnten unverwechselbare Marker schaffen, die eine eindeutige Markerzuordnung ermöglichen würden. Allerdings ist dies wie in Kapitel 2 kurz erwähnt nicht umzusetzen: um Informationen aus einem Marker beispielsweise in Form eines QR-Codes aus großen Höhen des Kites zu extrahieren, müssten diese Marker in nicht mehr praktikable Größenordnungen skaliert werden.

Aus diesem Grund wird ein semi-automatisierter Ansatz gewählt, der die Vorteile einer vom Benutzer manuell gesteuerten Berechnung mit denen einer vollautomatischen Lösung vereint. Dazu wählt der Nutzer zu Beginn sichtbare Marker aus, vergibt eindeutige IDs, die eine Zuordnung der Bildposition zu den realen Markerpositionen ermöglichen und startet anschließend die Berechnung (siehe Kapitel 5). Ist die Berechnung für den aktuellen Frame erfolgreich, werden die zuvor ausgewählten Marker automatisiert im nächsten Frame über die Berechnung des optischen Flusses verfolgt bzw. erkannt und die Pose berechnet.

Dabei hat jedoch der Nutzer in jedem Frame die Möglichkeit die Berechnung zu unterbrechen, wenn beispielsweise ein falsches Merkmal als Marker erkannt wird oder aber ein zuvor festgelegter Marker nicht korrekt über mehrere Frames verfolgt wurde. Unterbricht der Nutzer die Ausführung, kann im aktuellen Frame die Position bereits vorhandener Marker angepasst werden und die Berechnung mit den korrigierten Markern fortgesetzt werden. Darüber hinaus können jederzeit neue Marker hinzugefügt oder vorhandene entfernt werden. So kann bei Bedarf insbesondere auf plötzlich auftretende Umwelteinflüsse wie z. B. stark schwankender Lichteinfall oder nicht länger im Bild sichtbare Marker manuell reagiert werden.

Neben einer manuellen Unterbechung wird die automatisierte Berechnung auch beim Eintreten bestimmter Ereignisse automatisch unterbrochen. Am wichtigsten ist dabei der offensichtliche Fall, dass keine ausreichende Anzahl an Markern (siehe Kapitel 4.3) mehr im Bild sichtbar ist und die Berechnung logischerweise nicht fortgesetzt werden kann. In diesem Fall wird die Ausführung solange unterbrochen, bis der Nutzer eine ausreichende Anzahl an sichtbaren Markern auswählt.

4 Theoretische Erläuterung

Die Rekonstruktion der Pose basierend auf 2D-Aufnahmen ist elementarer Bestandteil dieser Arbeit. Allgemein ist es dabei Ziel, basierend auf 2D-Aufnahmen die relative Position und Orientierung der Kamera mit Referenz zu einem definierten Koordinatensystem zu bestimmen.

Dabei gliedert sich der Vorgang in verschiedene Teilprobleme, die im Folgenden theoretisch erläutert werden. Dazu bietet es sich an, zunächst die Teilprobleme zu extrahieren und im Anschluss die für jeden Teil relevanten Algorithmen vorzustellen. Dabei liegt der Fokus auf der Erläuterung der Grundidee und Funktionsweise der Algorithmen und nicht auf den in der Literatur hinreichend erläuterten mathematischen Herleitungen. Mathematische Darstellungen der Algorithmen finden sich beispielsweise in den jeweils angegebenen Quellen.

Grundsätzlich ist eine kalibrierte Kamera Voraussetzung für die Bestimmung der Pose für den gewählten Ansatz, da die bei der Kalibrierung bestimmten intrinsischen Kameraparameter für die Berechnung benötigt werden. Wenngleich dieser Vorgang je nach Auffassung als separate Vorbereitung und nicht Teil der eigentlichen Posenbestimmung gesehen werden kann, wird aus Gründen der Vollständigkeit zumindest grundlegend an dieser Stelle darauf eingegangen (siehe Kapitel 4.1).

Unter Verwendung einer kalibrierten Kamera kann dann die Pose allgemein über eine definierte Anzahl von 2D-3D-Punktkorrespondenzen bestimmt werden. Bei diesen Punkten handelt es sich um Merkmale im Bild, für die sowohl die aktuelle Bildposition als auch die reale 3D-Position bekannt sein müssen. Deshalb ist es zunächst notwendig, für die Berechnung der Pose die aktuelle Bildposition dieser Merkmale zu bestimmen. Dazu kommt ein Feature Detector zum Einsatz, der auffällige Merkmale des Bildes lokalisiert, zu denen die inneren Kanten der Marker zählen (siehe Kapitel 4.2). Anschließend kann die Pose mithilfe des in der Literatur als Perspective-n-Point-Problem (PnP) bezeichneten Ansatzes über die angesprochenden 2D-3D-Punktkorrespondenzen gelöst werden (siehe Kapitel 4.3).

4.1 Kamerakalibrierung

Die Kalibrierung einer Kamera ist erforderlich, um metrische Informationen aus einem 2D-Bild zu gewinnen. Dabei ist Ziel, eine Projektionsmatrix zu bestimmen, die eine perspektivische Abbildung definiert, die dreidimensionale Objektpunkte auf der entsprechenden 2D-Bildposition abbildet. Sie besteht aus extrinsischen Parametern, die die Kamerapose relativ zu ihrer Umgebung darstellen, und intrinsischen Parametern, die anhand des verwendeten Kameramodells die 3D-Punkte in homogene Bildkoordinaten transformieren (vgl. [Sim13]).

Bevor eine Berechnung der Pose stattfinden kann, ist deshalb die Bestimmung der intrinsischen Parameter erforderlich. Die intrinsischen Kameraparameter bei realen Kameras berücksichtigen Abweichungen beispielsweise durch Ungenauigkeiten in der Fertigung des Kamerasensors, die die Kamera eindeutig charakterisieren. Dazu zählen die Brennweite f_x und f_y, die tatsächliche Position des Hauptpunkts (engl. Principal Point) (c_x, c_y) sowie im Normalfall eine mögliche Scherung der Achsen, die jedoch von OpenCV vernachlässigt wird. Abb. 4 zeigt das von OpenCV verwendete Kameramodell. Dabei ist $(X, Y, Z)^T$ ein 3D-Punkt im Weltkoordinatensystem und (u, v) der zugehörige Bildpunkt.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$x'' = x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' = y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

$$\text{where} \quad r^2 = x'^2 + y'^2$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

Abbildung 4: Kameramodell von OpenCV mit intrinsischen Kameraparametern und Lens Distortion-Koeffizienten ([Ope15])

Darüber hinaus kommt es in Abhängigkeit von der verwendeten Kamera zu einer mehr oder weniger stark ausgeprägten Verzeichnung (engl. Lens Distortion). Dabei ändert sich mit zunehmendem Abstand von der optischen Achse die Vergrößerung zumeist radial und ist besonders präsent bei Weitwinkelaufnahmen. Bei der verwendeten GoPro-Kamera tritt radial eine sogenannte tonnenförmige Verzeichnung auf, bei der die Vergrößerung zum Bildrand abnimmt. Dieser Effekt wird ebenfalls im Rahmen einer Kalibrierung durch eine Erweiterung des verwendeten Kameramodells als Modellparameter berücksichtigt (vgl. [Ope15]). In Abb. 4 werden die radialen als k_i bezeichnet, während p_1 und p_2 tangentiale Lens Distortion darstellen. Dabei werden standardmäßig $k_1,\ k_2,\ k_3,\ p_1,\ p_2$ bei der Kalibrierung mit OpenCV bestimmt.

Der in OpenCV verwendete Ansatz basiert auf [Zha98], die eine Kamerakalibrierung basierend auf einem bekannten planaren Kalibrierobjekt zumeist in Form eines Schachbrettmusters einführen. Dazu nutzen sie mehrere Bilder des Kalibrierobjekts mit variierenden Orientierungen. Zunächst werden in jedem Bild die Position der inneren Ecken des Schachbrettmusters detektiert und Homographien zwischen den einzelnen Bildern berechnet. Basierend auf diesen Homographien werden Initialwerte für die intrinsischen $(f_x,\ f_y,\ c_x,\ c_y,\ s)$ und extrinsischen (Pose der Kamera) Kameraparametern bestimmt, bevor mithilfe einer Ausgleichsrechnung in Form der Methode der kleinsten Quadrate die Parameter zur Verzeichnung bestimmt werden. Abschließend werden alle

Initialwerte der Parameter durch Minimierung der Maximum-Likelihood-Methode verfeinert (vgl. [Zha98], S. 8).

Die so gewonnenen Kameraparameter werden insbesondere zum Verfolgen der Marker über mehrere Frames (Kapitel 4.2) sowie der Lösung des PnP-Problems (Kapitel 4.3) benötigt. Die Qualität der Kalibrierung bzw. die möglichst genaue Bestimmung der intrinsischen Parameter beeinflusst also direkt die Genauigkeit der Posenberechnung. Um eine Aussage über die Qualität der Kamerakalibrierung treffen zu könnnen, wird deshalb das quadratische Mittel des Reprojektionsfehlers für alle bei der Kalibrierung verwendeten Bildpunkte bestimmt und liefert einen Referenzwert in Pixeln. Um die Kalibrierung zu verbessern, sollte versucht werden, diesen so gering wie möglich zu halten.

4.2 Merkmalerkennung

Eine elementare Aufgabe in der Analyse und Auswertung von digitalen Bildern ist die Erkennung von distinktiven Merkmalen (Features) im Bild, die die Grundlage für weiterführende Aufgaben wie Objekterkennung und -tracking bilden. Die vorhandenen Ansätze zur elementaren Feature Detection lassen sich in drei Kategorien einteilen: Konturbasierte Methoden extrahieren zunächst Konturen im Bild und werten anschließend deren Krümmung aus, um Features zu erkennen. Im Gegensatz dazu nutzen intensitätsbasierte Verfahren die Grauwertintensität in Pixelregionen, die typischerweise an Eckpunkten stark variiert, wohingegen modellbasierte Ansätze zuvor definierte Modelle für spezifische Features zur Erkennung nutzen (vgl. [SMB00], S. 153).

Bei den konkret verwendeten Markern handelt es sich um Schachbrettmarker, die aus je zwei weißen und schwarzen angrenzenden Flächen bestehen (siehe Kapitel 3). Die Differenz der Grauwertintensität an den Übergängen dieser Flächen ist im Idealfall also maximal. Daher bieten sich zur Erkennung dieser Marker intensitätsbasierte Feature Detectors an. Im Rahmen dieser Arbeit kommen deshalb für die Implementierung der Posenberechnung der Harris Corner Detector, der wohl bekannteste in dieser Kategorie, sowie eine Weiterentwicklung davon durch Shi und Tomasi ([ST94]), der Good Features To Track-Ansatz, zum Einsatz.

Harris und Stephens ([HS88]) definieren basierend auf dem Gradientenverlauf in x- und y-Richtung um einen Pixel eine sogenannte Corner Response-Funktion, die jedem Pixel einen Wert zur Ausgeprägtheit einer möglichen Ecke an diesem Punkt zuweist. Dabei nutzen sie aus, dass an einem Eckpunkt die Gradienten in alle Richtungen starke Änderungen aufweisen. Dazu wird ein definierter Pixelbereich zumeist in geglätteter runder Form (z. B. mittels Gauß-Filter) in der Nachbarschaft des Pixels betrachtet, für den die Eigenwerte der Autokorrelationsmatrix und daraus die Corner Response

des Pixels berechnet wird. Die Corner Response Function ist dabei so definiert, dass eine negative Ausgabe Indiz für eine Kante ist, wohingegen positive Werte eine Ecke beschreiben. Für homogene Bildregionen sind kleine Werte zu erwarten (vgl. [HS88], S. 149 f.).

Derselbe prinzipielle Ansatz zur Erkennung von Ecken wurde von Shi und Thomasi aufgegriffen, die jedoch die Corner Response-Funktion veränderten. Während die originale Corner Response-Funktion für eine Ecke voraussetzt, dass die beiden Eigenwerte λ_1 und λ_2 der Autokorrelationsmatrix große Werte annehmen und nicht $\lambda_1 << \lambda_2$ bzw. $\lambda_1 >> \lambda_2$ (vgl. [HS88], S. 149 f.), setzen Shi und Thomasi lediglich voraus, dass beide Eigenwerte oberhalb des Schwellwertes λ liegen ([ST94], S. 595).

Um Marker in einer Sequenz von Bildern automatisiert zu erkennen, wie im Rahmen dieser Arbeit benötigt (siehe Kapitel 5.3), bedarf es hingegen mehr als das elementare Erkennen der Features in einem Bild. Es muss darüber hinaus Mechanismen zur Wiedererkennung derselben Merkmale in nachfolgenden Bildern geben. Es muss also die Bewegung der Objekte im Bild verfolgt werden, was der Berechnung des optischen Flusses (engl. Optical Flow) entspricht. Einen solchen Algorithmus stellt die Lucas-Kanade-Methode dar. Ziel ist es, zu einem in Bild I bekannten Punkt $u = [u_x \ u_y]^T$ in einem zweiten Bild J den Punkt $v = u + d = [u_x + d_x \ u_y + d_y]^T$ zu finden (vgl. [Bou00], S. 1). v in Bild J entspricht dabei demselben Merkmal wie in Bild I jedoch nach einer Bewegung. Dabei ist d der optische Fluss. Die Kernidee des Algorithmus von Lucas und Kanade ist, dass der optische Fluss in einer lokalen Umgebung um einen Pixel konstant ist. Unter dieser Annahme setzt sich das Problem zusammen aus dem Registrieren von Features und der anschließenden Berechnung des optischen Flusses basierend auf den partiellen Ableitungen für diese Features.

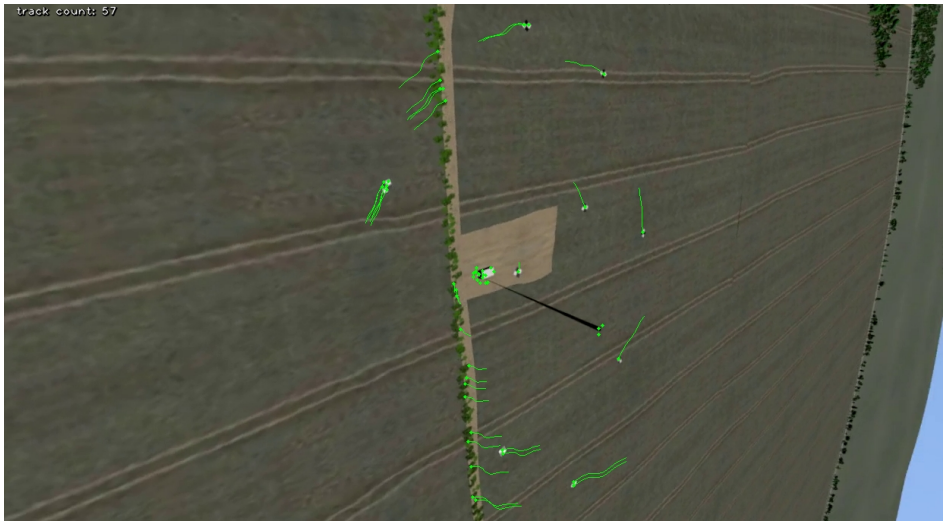


Abbildung 5: Visualisierung des optischen Flusses als grüner Schweif über zehn Frames in einem Ausschnitt der Simulation während einer Drehung des Kites

Bouguet ([Bou00]) beschreibt dabei eine pyramidale Umsetzung, die zum Tracken der Marker in dieser Arbeit in OpenCV zur Anwendung kommt. Dazu wird das Bild I rekursiv als Pyramide aufgebaut, wobei das erste Level der Pyramide das Originalbild darstellt und jedes weitere Level kleiner werdende Ausschnitte des Originals enthält. Anschließend wird für das unterste Level der Pyramide bzw. den kleinsten Bildausschnitt der optische Fluss basierend auf der Annahme, dass dieser in einer lokalen Umgebung nahezu konstant ist, berechnet. Diese Berechnung wird als initialer Schätzwert dem nächsthöheren Level zur Verfügung gestellt, wo der optische Fluss auf Basis des Schätzwerts verfeinert wird. So durchläuft der Algorithmus alle Level der Pyramide und liefert abschließend den optischen Fluss im Originalbild (vgl. [Bou00], S. 1 f.). Abb. 5 visualisiert den mithilfe des Ansatzes von Bouguet berechneten optischen Fluss über zehn Frames.

Da die Bildposition der Marker direkt die Berechnung der Pose beeinflusst, ist es entscheidend, ihre Position so genau wie möglich zu bestimmen. Dazu bietet es sich an, die bestimmten Bildpositionen auf Subpixelgenauigkeit zu optimieren. Während der Lucas-Kanade-Feature Tracker diese Optimierung bereits über eine biliniare Interpolation enthält ([Bou00], S. 7), bietet OpenCV die Möglichkeit zuvor erkannte Features nachträglich manuell zu verfeinern. Dazu kommt ein Algorithmus zum Einsatz, der auf zuvor mit Pixelgenauigkeit bestimmte Eckpunkte angewendet werden kann, um diese zu verfeinern. Der Algorithmus basiert dabei auf der Beobachtung, dass jeder Vektor ausgehend von einem Eckpunkt q zu einem Punkt p in einer definierten Umgebung von q orthogonal zum Gradienten an Punkt p ist. (vgl. [Ope15]).

4.3 Perspective-n-Point-Problem

Das PnP-Problem beschäftigt sich mit der Bestimmung der Pose einer kalibrierten Kamera basierend auf n Kontrollpunkten für die sowohl die reale 3D-Position als auch die Bildposition bekannt sind (vgl. [LMNF09], S. 1). Bei den Kontrollpunkten handelt es sich beispielsweise um markante Punkte eines Objekts oder in diesem Fall um innere Kanten der Marker. Dabei soll die relative Position zwischen der Kamera und der Umgebung gefunden werden.

Das PnP-Problem kann als die Bestimmung der Längen der Teilstrecken, die das Center of Perspective (CP) mit den einzelnen Kontrollpunkten verbinden, beschrieben werden (vgl. [FB81], S. 383). Beim CP handelt es sich um einen zusätzlichen Punkt, der der Öffnung im Modell der Lochkamera entspricht, durch die die Strahlen einfallen. Zur Berechnung der Teilstrecken werden jeweils die Winkel zwischen zwei Kontrollpunkten ausgehend vom CP betrachtet.

Dabei besteht ein wesentliches Merkmal des PnP-Problems darin, dass die Anzahl vorhandener Kontrollpunkte maßgeblich über die Anzahl möglicher Lösungen entscheidet.

Für $n = 1$ und $n = 2$ gibt es jeweils unendlich viele Lösungen. Von besonderem Interesse ist der Fall $n = 3$, da es sich beim P3P-Problem um den ersten Fall mit einer endlichen Anzahl von Lösungen handelt, der zudem immer einen Spezialfall bei PnP-Problemen mit $n > 3$ darstellt (vgl. [GHTC03], S. 3). Für $n = 3$ beträgt die maximale Anzahl physikalisch sinnvoller Lösungen vier. Als physikalisch sinnvoll werden dabei Lösungen definiert, die für den Anwendungsfall tatsächlich relevant sind ([GHTC03], S. 5). Keine physikalische Lösung wäre somit z. B. eine Lösung, bei der die berechneten Punkte hinter der Kamera und somit nicht im Bild liegen.

Daher ist die Annahme naheliegend, dass sich auch bei der Posenbestimmung des fliegenden Kites immer mindestens drei Marker im Sichtfeld der Kamera befinden müssen, deren Position im Bild zudem korrekt erkannt wurde, um vier mögliche Posen bestimmen zu können. Allerdings wird dies durch OpenCV beschränkt, da dort für alle implementierten Algorithmen des PnP-Problems mindestens vier Punkte vorausgesetzt werden, um die Pose eindeutig bestimmen zu können. Somit werden zur Posenberechnung basierend auf OpenCV immer vier sichtbare Marker benötigt.

Die Lösung des PnP-Problems liefert dabei nicht direkt die Pose der Kamera, sondern die Pose des Objekts, zu dem die verwendeten Kontrollpunkte gehören. Daraus kann jedoch ohne großen Aufwand die Pose der Kamera als Inverse der ermittelten Transformationsmatrix bestimmt werden.

Zu beachten ist dabei, dass die Distanz zwischen den Markern bzw. die von den Markern eingeschlossene Fläche für größere Abstände bzw. Flächen theoretisch genauer ist. Dies ist einfach nachzuvollziehen, da bei relativ kleinen Abständen zwischen den Markern auch relativ kleine Abweichungen der ermittelten Markerposition größere Auswirkungen auf Fehler in der Pose haben, als es bei größeren Entfernungen und somit, bei gleichbleibendem Messfehler der Markerposition, kleineren relativen Abweichungen der Fall wäre.

Dabei bietet es sich als gutes Fehlermaß an, im Anschluss an die Lösung des PnP-Problems den Reprojektionsfehler zu bestimmen, was im Rahmen dieser Arbeit ebenfalls umgesetzt wurde. Dazu wird basierend auf der zuvor ermittelten Pose und der bekannten 3D-Position der verwendeten Kontrollpunkte bzw. Marker ihre Position im Bild berechnet und die Abweichung zur im Bild erkannten Markerposition bestimmt.

Heutzutage existiert eine Vielzahl von Algorithmen zur Lösung des PnP-Problems, die zumeist auf eine effiziente Berechnung sowie Robustheit ausgelegt sind. Im Folgenden sollen die in OpenCV zu diesem Zeitpunkt implementierten Algorithmen kurz vorgestellt werden.

Bei der OpenCV-Standardmethode wird unter Verwendung des Levenberg-Marquardt-Algorithmus der Reprojektionsfehler minimiert. Konkret werden dabei die Summen der quadrierten Distanzen zwischen den bekannten Bildpunkten und projizierten Bildpunkten iterativ minimiert (vgl. [Ope15]).

Ein weiterer in OpenCV implementierter Ansatz beschäftigt sich ausschließlich mit den möglichen Lösungen des P3P-Problems. Die Implementierung in OpenCV setzt für diesen Algorithmus trotzdem immer genau vier Kontrollpunkte voraus. Der Algorithmus nach [GHTC03] berechnet dabei abhängig von der Konstellation der Kontrollpunkte bis zu vier Lösungen. Um daraus eine eindeutige Lösung zu bestimmen, werden Kriterien definiert, die das P3P-Problem in Abhängigkeit der Lösungsanzahl komplett klassifizieren. Die zugehörigen Kriterien sind ebenfalls in [GHTC03] definiert.

Daneben steht der sogenannte EPnP-Ansatz zur Verfügung, der einen nicht-iterativen Ansatz darstellt, dessen Zeitkomplexität mit $\mathcal{O}(n)$ geringer ist als bei anderen modernen Ansätzen (vgl. [LMNF09], S. 155). Dabei wird nicht wie bei vielen anderen Ansätzen versucht, das PnP-Problem über die Tiefe der Kontrollpunkte zu lösen, sondern stattdessen die Kontrollpunkte als gewichtete Summe von vier virtuellen Punkten darzustellen. Die Unbekannten sind dann nicht länger die Kontrollpunkte, sondern die vier virtuellen Punkte. So kann insbesondere bei einer hohen Zahl von Kontrollpunkten die Laufzeit weiter gering gehalten werden (vgl. [LMNF09], S. 158)

Diese Ansätze haben jedoch gemeinsam, dass sie empfindlich gegenüber Ausreißern sind. Abhilfe schafft der Random Sample Consensus (RANSAC)-Ansatz, der insbesondere bei $n > 4$ relevant wird. Dieser Ansatz wählt zufällig die minimale Anzahl an Daten aus, die nötig sind, um ein gegebenes Modell zu instanziieren. In diesem Fall ist dies die Lösung des PnP-Problems, das wie zuvor erläutert mindestens vier Werte bzw. Punkte zur Berechnung benötigt. Anschließend wird mit dieser minimalen Auswahl das Modell instanziiert bzw. im konkreten Fall die Pose berechnet. Anhand der ermittelten Lösung wird dann über einen definierten Schwellwert geprüft, wie gut die anderen $n-4$ Punkte in das Modell passen. Liegt eine zuvor definierte Anzahl an Daten innerhalb des Schwellwertes wird das Modell mit diesen Werten optimiert, ohne dass Werte außerhalb des Schwellwerts in die Berechnung einbezogen werden (vgl. [FB81], S. 383 f.). So haben auch extreme Ausreißer keinen Einfluss mehr auf das endgültige Ergebnis. Liefert das gewählte Modell hingegen keine ausreichende Anzahl an Punkten innerhalb des Schwellwertes, wird erneut ein zufälliger minimaler Datensatz ausgewählt und der Prozess beginnt erneut, bis ein passender Datensatz gefunden oder eine bestimmte Anzahl an Iterationen überschritten wird und der Algorithmus terminiert. Die Lösung des PnP-Problems unter Verwendung von RANSAC ermöglicht es so, die zuvor erwähnten Lösungsansätze unempfindlich gegenüber Ausreißern zu machen.

Die verschiedenen Algorithmen werden im Rahmen der Evaluation eingesetzt und unter verschiedenen Bedingungen und Parameterisierungen getestet (siehe Kapitel 7).

5 Implementierung

Die Software, die im Rahmen dieser Arbeit entwickelt wurde, ist unter Verwendung des Build Systems CMake in Version 2.8.12 in C/C++ geschrieben und verwendet die quelloffene Bildverarbeitungsbibliothek OpenCV in Version 2.4.10. OpenCV bietet eine ausgereifte Implementierung für eine Vielzahl von Algorithmen insbesondere aus dem Bereich der Bildverarbeitung und den mathematischen Teilgebieten der linearen Algebra und Geometrie. Dabei wird OpenCV durch eine große Entwicklergemeinde ständig erweitert.

Die Kalibrierung der Kamera erfolgt zunächst in einem separaten Programm, das auf der OpenCV-Demonstration zur Kamerakalibrierung basiert und im Rahmen dieser Arbeit nur geringfügig modifiziert wurde, um zusätzlich eine Konfigurationsdatei für das Programm zur Posenbestimmung zu erzeugen, die unter anderem die Kameramatrix und realen Markerpositionen enthält. Der Kern dieser Arbeit ist die Umsetzung des Programms zur Posenbestimmung, das im Folgenden erläutert wird.

5.1 Programmstruktur

Das entwickelte Programm implementiert die Posenberechnung in Form eines Zustandsautomaten, der externe Ereignisse (Events) in Form von Nutzereingaben sowie innerhalb der Zustände generierte interne Ereignisse verarbeitet. Da dies jedoch nicht weiter zur Posenberechnung beiträgt, wird darauf an dieser Stelle nicht weiter eingegangen. Eine Übersicht der verschiedenen Zustände und Zustandsübergängen befindet sich im digitalen Anhang.

Im Folgenden wird zunächst eine grobe Übersicht des Programmablaufs aufgezeigt, bevor einzelne Teile hervorgehoben und erläutert werden.

Zunächst erfolgt die Initialisierung, bei der die benötigten Datenstrukturen angelegt und bei Bedarf initialisiert werden. Dazu gehören neben den bei der Kamerakalibrierung bestimmten Parametern (Kameramatrix und Verzerrungskoeffizienten) insbesondere das Video des Kiteflugs. Dabei wird darauf verzichtet, das Video in seiner Gesamtheit in den Arbeitsspeicher zu laden. Stattdessen stehen immer nur der aktuelle sowie der vorherige Frame, der zum Tracken der Marker benötigt wird (siehe Kapitel 5.3), als OpenCV-Matrix zur Verfügung. Für jeden Frame wird jedoch ein für diese Arbeit entwickeltes Objekt erzeugt, das zur Laufzeit für die Posenberechnung relevante Daten erfasst. Dazu gehören unter anderem die im jeweiligen Frame ausgewählten Marker und, falls die Berechnung der Pose bereits stattgefunden hat, die entsprechenden Werte der Pose. In Abb. 6 ist diese Datenstruktur als Frameinformationen bezeichnet und zeigt die Interaktion bei der Posenberechnung. Dabei gibt sie keine Auskunft über Videoinformationen des Frames, sondern beinhaltet für jeden Frame die oben erwähnten

Daten. Durch die Informationen über die einzelnen Frames kann bei der Programmnutzung zwischen bereits berechneten Frames gewechselt und nachträglich Änderungen an der Markerauswahl etc. vorgenommen werden, ohne unnötig Speicher mit Matrizen der Frames des Videos zu belegen. Darüber hinaus steht eine Übersicht der vorhandenen Marker zur Verfügung (Markerinformationen in Abb. 6), die im Verlauf den aktuellen Zustand der Marker erfasst und ihre reale Position enthält (siehe Kapitel 5.3). Die Markeranzahl und 3D-Position wird dazu aus einer bei der Kalibrierung erstellten XML-Konfigurationsdatei eingelesen.

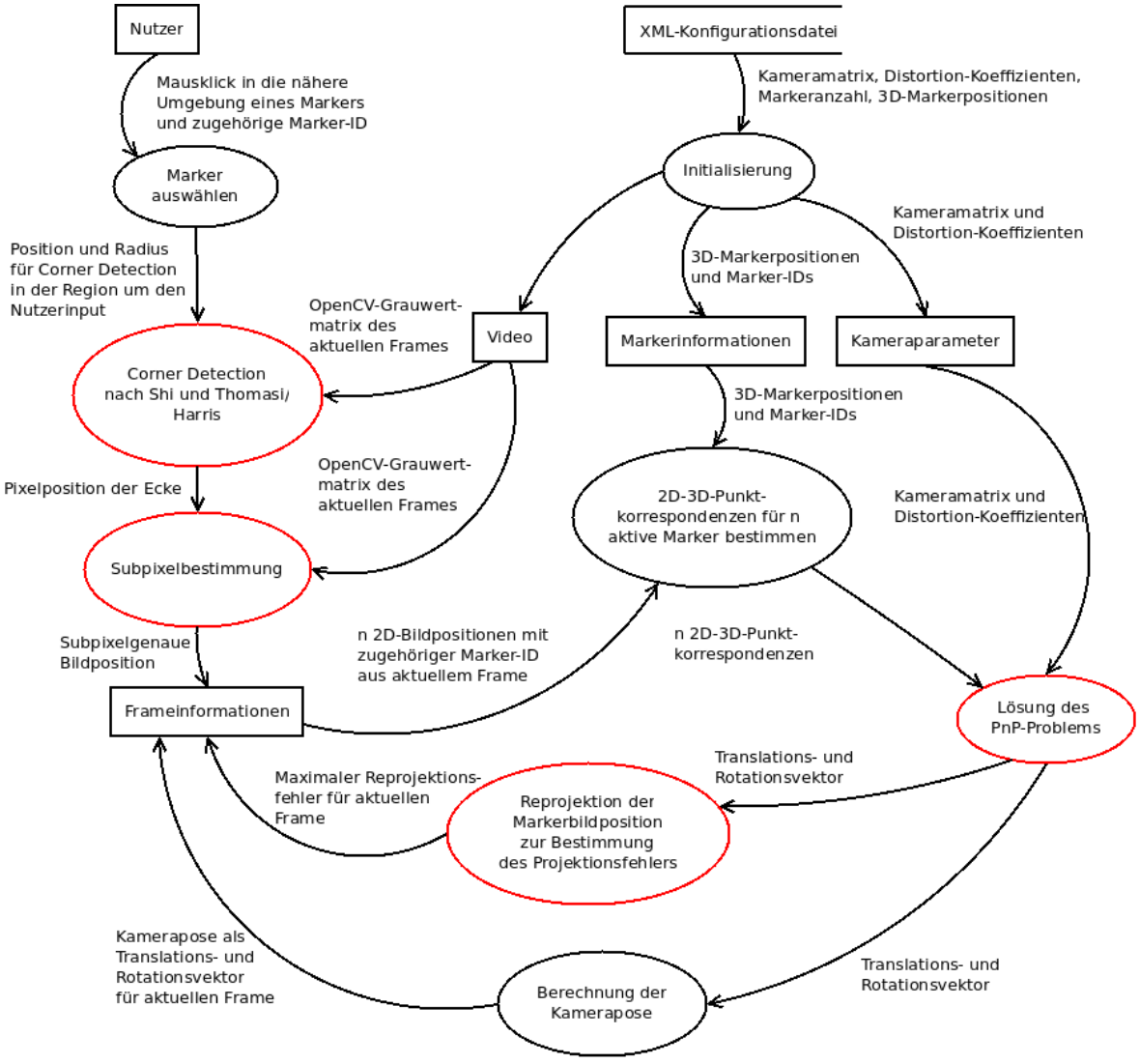


Abbildung 6: Datenfluss der Markerauswahl und allgemeinen Posenberechnung. OpenCV-Funktionen werden in rot dargestellt

Anschließend müssen mindestens vier Marker ausgewählt werden (siehe Kapitel 5.2). Je nach Bedarf kann dann die Berechnung der Pose nur für den aktuellen Frame gestartet oder automatisiert über nachfolgende Frames bestimmt werden, bis die Berechnung durch eine Nutzereingabe oder ein internes Event unterbrochen wird. Abb. 6 zeigt eine Übersicht der Initialisierung, Markerauswahl und Posenberechnung für einen Frame. Die Markerauswahl muss dabei zunächst für mindestens vier Marker sequenziell er-

folgen, bevor die Posenberechnung gestartet werden kann. Zu beachten ist, dass aus Gründen der Übersichtlichkeit das Tracken der Marker zunächst in dieser Übersicht nicht dargestellt, sondern separat behandelt wird.

5.2 Markerauswahl

Die umgesetzte Markerauswahl bzw. -erkennung kombiniert Nutzereingaben mit Algorithmen zur Merkmalerkennung. Zunächst muss bei der entsprechenden Eingabe des Nutzers zum Hinzufügen eines Markers die entsprechende Identifikation des Markers eingegeben werden, die bereits beim Platzieren der Marker vor den Videoaufnahmen festgelegt werden musste. An dieser Stelle sei erwähnt, dass zur Realisierung jeglicher Nutzereingaben die primitive, von OpenCV bereitgestellte Benutzerschnittstelle genutzt wird. Diese ermöglicht es, auf einzelne Tastatureingaben in Form von ASCII-Zeichen zu warten, solange das entsprechende OpenCV-Fenster im Vordergrund aktiv ist. Wenngleich dieser Ansatz für den Rahmen dieser Arbeit ausreichend ist, bringt sie doch einige Beschränkungen mit sich. So führt beispielsweise die Eingabe eines nicht ASCII-konformen Zeichens standardmäßig zum Absturz des Programms durch OpenCV.

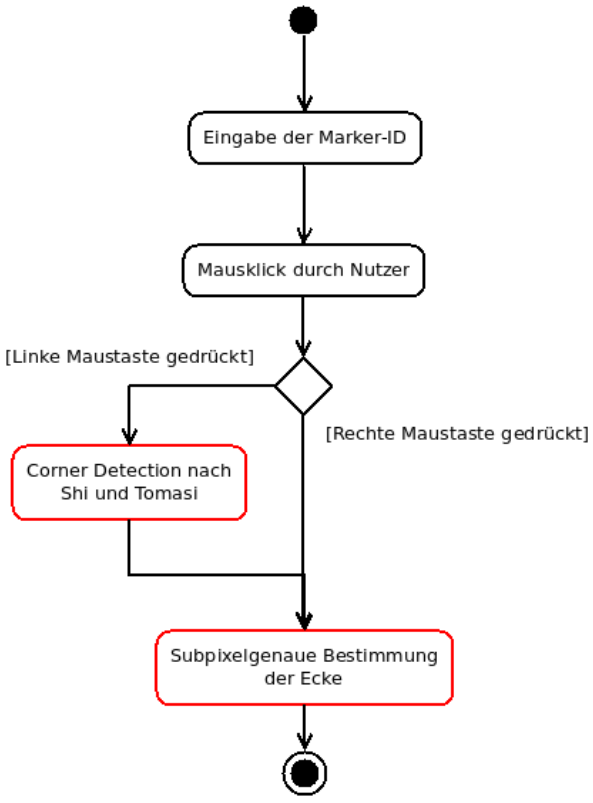


Abbildung 7: Zwei Möglichkeiten zur Markerauswahl mit linker und rechter Maustaste. OpenCV-Funktionen werden in rot dargestellt.

Bei Eingabe einer gültigen Marker-ID, hat der Nutzer zwei Möglichkeiten zur Auswahl des Markers im Bild, die in Abb. 7 dargestellt sind: Bei einem Klick mit der linken Maustaste wird in einem definierten Bereich um die aktuelle Position der Maus die innere Kante des Markers pixelgenau detektiert. Dazu wird eine kreisförmige Maske um die Position des Klicks gelegt, bevor einer der im theoretischen Teil erläuterten OpenCV-Algorithmen zur Detektion angewendet wird. Die Größe dieser Maske lässt sich in der Konfigurationsdatei für die Posenberechnung festlegen. Zur besseren Übersicht für den Nutzer wird bei der Markerauswahl der kreisförmige Bereich um den Mauszeiger farblich hervorgehoben, in dem der Algorithmus zur Merkmaldetektion im Falle eines Klicks zur Anwendung kommen würde.

Da jedoch durch äußere Einflüsse nicht immer gewährleistet werden kann, dass die Merkmalerkennung korrekt funktioniert, besteht darüber hinaus die Möglichkeit, manuell einen Pixel mit der rechten Maustaste auszuwählen. Dies ist komfortabel möglich, da mit OpenCV dargestellte Frames standardmäßig über die Funktion verfügen, mithilfe des Mausrads bis auf die Pixelebene zu vergrößern. Aus diesem Grund empfiehlt es sich, für die Bedienung des Programms eine Drei-Tasten-Maus zu verwenden. Für die Auswertung der Interaktion mit der Maus wurde eine Callback-Funktio definiert, die die jeweilige Maustaste und Bewegung der Maus über dem dargestellten Frame bestimmt und der Posenbestimmung zur Verfügung stellt.

Unabhängig von der initialen Markerdetektion mit rechter oder linker Maustaste wird im Anschluss die Position auf Subpixelgenauigkeit mit der entsprechenden OpenCV-Funktion optimiert und der Posenberechnung zur Verfügung gestellt.

In realen Aufnahmen ist ein Teil des Kontrollsockels unterhalb des Kites im Bild sichtbar. Aus diesem Grund wird eine Region im Bild definiert, in der keine Marker vom Nutzer ausgewählt werden können oder automatisiert verfolgt werden. Dazu wird eine Maske auf das Bild gelegt und die entsprechende Region ignoriert.

5.3 Markertracking

Zur automatisierten Posenberechnung ist es erforderlich, die vom Nutzer ausgewählten Marker über nachfolgende Frames zu verfolgen. Dazu muss bekannt sein, welche Marker aktuell zum Tracken verwendet werden sollen. Dies ist wie bereits erwähnt anhand der globalen Markerübersicht, die kontinuierlich aktualisiert wird, einzusehen.

Zur Nutzung des Lucas-Kanade-Algorithmus ist es erforderlich, Grauwertbilder für den vorherigen und aktuellen Frame zur Verfügung zu stellen. Diese können aus den entsprechenden Matrizen des Videos gewonnen werden. Der generelle Ablauf der Markerverfolgung für einen Frame i wird in Abb. 8 dargstellt.

Um auch kurzzeitig verdeckte oder nicht korrekt erkannte Marker wiedererkennen zu können, werden diese nicht sofort aus der Liste der aktiven Marker entfernt, sondern

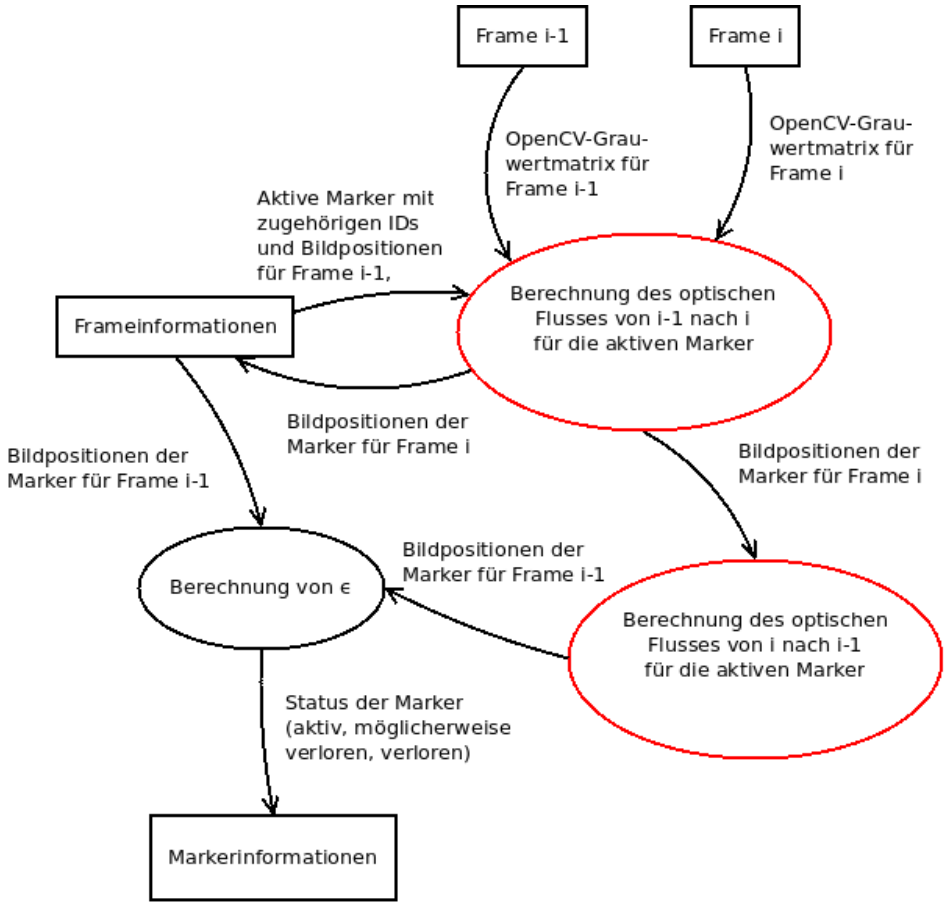


Abbildung 8: Markerverfolgung von Frame i-1 zu Frame i

stattdessen mit einem Flag als möglicherweise verloren gekennzeichnet. Dabei ist es erforderlich, zu definieren, wann Marker als möglicherweise verloren anzusehen sind. Für diese Arbeit wird dazu zunächst mithilfe des optischen Flusses für die Marker ausgehend vom vorherigen Frame $i-1$ von Position $u_{i-1,orig}$ zum aktuellen Frame i die neue Position $u_{i,estimate}$ bestimmt. Im Anschluss werden die neu berechneten Markerpositionen in Frame i als Ausgangspunkte gewählt und stattdessen der optische Fluss genutzt, um die Position $u_{i-1,estimate}$ im vorherigen Frame zu berechnen. Als Qualitätsmaß der neu berechneten Markerposition wird dann die Differenz zwischen der urprünglichen Markerposition $u_{i-1,orig}$ und der basierend auf der neu berechneten Position $u_{i,estimate}$ zurückgerechneten Position $u_{i-1,estimate}$ verwendet, die möglichst klein sein sollte. Das Kriterium für eine korrekt verfolgte Markerposition $u_{i,estimate}$ wird somit definiert als:

$$\|u_{i-1,orig} - u_{i-1,estimate}\| < \epsilon$$

Dabei ist ϵ ein Schwellwert in Pixeln, der bei Bedarf angepasst werden kann und im Programm standardmäßig als $\epsilon = 0,5\ Px$ definiert ist. Übersteigt eine ermittelte Markerposition den Schwellwert, so wird der Marker als möglicherweise verloren gekennzeichnet. Diese Marker werden dann nicht länger als rote, sondern blaue Kreuze

dargestellt, bis sie wieder erkannt oder vollständig entfernt werden.

Um diese Marker in den nachfolgenden Frames wiederzuerkennen, wurden zwei Ansätze implementiert, deren jeweiliger Ablauf in Abb. 9 dargestellt wird: Sofern neben einem oder mehreren möglicherweise verlorenen Markern mindestens vier weiterhin korrekt erkannt wurden, bietet es sich an, basierend auf den sichtbaren Markern die aktuelle Pose der Kamera zu berechnen (siehe Kapitel 4.3). Anschließend können, sofern der Reprojektionsfehler innerhalb des definierten Bereichs liegt, für die möglicherweise verlorenen Marker aus ihrer bekannten 3D-Position und der Kamerapose ihre aktuelle Bildposition bestimmt werden und die Marker in folgenden Frames wieder für die Posenberechnung genutzt werden.

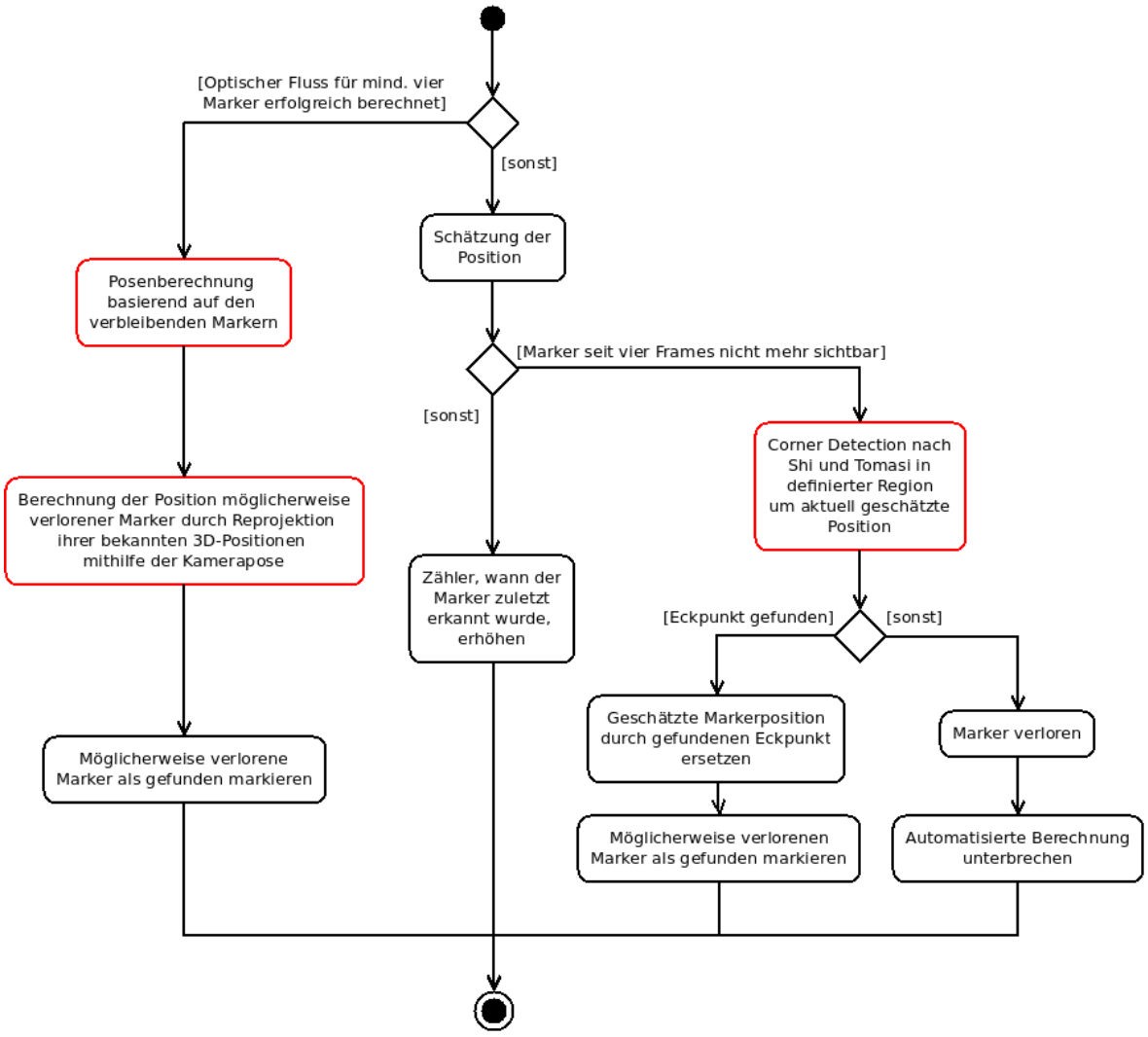


Abbildung 9: Ablauf für jeden nachfolgenden Frame, sobald mindestens ein Marker als möglicherweise verloren markiert wurde.

Für den Fall, dass nur vier Marker aktiv sind, wird davon ausgegangen, dass der wahrscheinlichste Grund für einen verdeckten Marker das Seil des Kites, insbesondere während einer Drehung, ist. Dabei wird die Annahme getroffen, dass das Seil den Mar-

ker nur für wenige Frames überdeckt. Aus diesem Grund wird für den Fall $n = 4$ aktive Marker bei einer Verdeckung von Markern von einer linearen Bewegung der Marker ausgegangen, die basierend auf den Bildern i und j berechnet wird. Frame i ist dabei der Frame, in dem der Marker zum vorletzten Mal korrekt erkannt wurde, und j der aktuelle Frame. Dass Frame i nicht dem letzten Frame entspricht, in dem der Marker erkannt wurde, liegt daran, dass eine Evaluierung der Funktionalität schnell gezeigt hat, dass das Seil im Frame vor dem Verschwinden eines Markers bereits Einfluss auf die mittels Lucas-Kanade-Algorithmus berechnete Position hat. Das Seil verschiebt in dem Frame die erkannte Position bereits leicht in die Richtung, in die das Seil sich bewegt, wenngleich die ermittelte Position noch innerhalb des Schwellwertes liegt. Um diesen unerwünschten Einfluss zu verhindern, wird stattdessen der vorletzte Frame verwendet.

Da die Annahme einer linearen Bewegung nur grob näherungsweise und über kurze Zeiträume korrekt ist, wird definiert, dass das Seil nach vier Frames den Markern wieder verlassen haben sollte. Deshalb wird vier Frames nach dem Verschwinden des Markers die aktuelle geschätzte Position des Markers als Ausgangspunkt für eine Merkmalerkennung mit dem Ansatz von Shi und Tomasi ([ST94]) innerhalb eines definierten Radius genutzt, der standardmäßig in der Konfigurationsdatei auf 15 Px eingestellt wird. Wird innerhalb dieses Bereichs ein Eckpunkt erkannt, wird davon ausgegangen, dass es sich um den vermissten Marker handelt und seine Position entsprechend angepasst. Der Marker wird wieder als erkannt markiert und eine Posenberechnung durchgeführt. Dabei ist die Merkmalerkennung, abhängig von der tatsächlichen Markerbewegung, fehleranfällig. Sollte der Marker jedoch tatsächlich an eine falsche Position gesetzt werden, so würde die anschließende Posenberechnung für den Marker einen großen Reprojektionsfehler liefern und eine weitere Berechnung stoppen (siehe Kapitel 5.4), bis eine Korrektur durch den Nutzer erfolgt. Gleiches gilt, wenn kein Merkmal innerhalb des Radius gefunden wird. Auch in diesem Fall wird die weitere Berechnung unterbrochen und dem Nutzer die Möglichkeit zur Interaktion gegeben.

Eine lineare Schätzung liefert jedoch keine qualitativen Messwerte, sodass häufig größere Fehler in der bestimmten Pose die Folge sind. Dies kann jedoch entweder durch die Verringerung des maximal zulässigen Reprojektionsfehlers in der Konfigurationsdatei oder einer nachträglichen Filterung der Messwerte basierend auf den jeweiligen Reprojektionsfiltern verhindert werden.

Alternativ bietet es sich, sofern vorhanden, an, mehr als vier Marker auszuwählen, da die automatisierte Berechnung dann deutlich robuster ist und, sofern RANSAC zur Anwendung kommt, krasse Ausreißer problemlos gefiltert werden können, ohne die Ausführung zu unterbrechen.

Daneben kann es passieren, dass Marker das Bild verlassen und nicht länger sichtbar sind. In diesem Fall liegt ihre projizierte Bildposition außerhalb der Dimsensionen

des Bildes. Dann werden die Marker als außerhalb des Bildes liegend markiert und nicht länger für die Berechnung der Pose verwendet. Allerdings wird weiterhin bei einer ausreichenden verbleibenden Anzahl an Markern die Pose automatisiert berechnet und die Bildposition der außerhalb des Bildes liegenden Markern reprojiziert. Taucht ein solcher Marker wieder innerhalb des Bildes auf, wird die reprojizierte Position als aktuelle Position des Markers definiert und der entsprechende Marker in nachfolgenden Frames erneut bei der Posenberechnung mit einbezogen. Dabei erfolgt das Erkennen dieser Marker analog zum linken Zweig in Abb. 9 für verlorene Marker.

Dabei wird der Status aller Marker zurückgesetzt, sobald das automatisierte Verfolgen durch den Nutzer oder durch ein Ereignis wie beispielsweise einer unzureichenden Anzahl an verbleibenden Markern unterbrochen wird.

5.4 Posenberechnung

Die Posenberechnung bzw. Lösung des PnP-Problems kann durch den Nutzer automatisiert oder für den aktuellen Frame gestartet werden, wenn mindestens vier Marker durch den Nutzer ausgewählt wurden. Dann wird zunächst aus den ständig aktualisierten Markerinformationen und den Informationen des aktuellen Frames die reale sowie 2D-Bildposition der Marker ausgelesen. Daneben benötigt der Algorithmus die intrinsischen Kameraparameter und Verzerrungskoeffizienten, die bereits bei der Initialisierung aus der Konfigurationsdatei ausgelesen wurden.

Dabei wird bei der Lösung des PnP-Problems nicht direkt die Pose der Kamera relativ zum Weltkoordinatensystem bestimmt. Vielmehr wird die Rotation und Translation des Objekts berechnet, zu dem die Kontrollpunkte gehören und diese im lokalen Koordinatensystem der Kamera dargestellt. Um daraus die Pose der Kamera zu bestimmen, ist es nötig, die ermittelte Transformationsmatrix zu invertieren. Dazu wird zunächst der bestimmte Rotationsvektor mithilfe der Rodrigues-Formel als Rotationsmatrix dargestellt. Diese bildet zusammen mit dem Translationsvektor die Transformationsmatrix und muss anschließend invertiert werden. Abb. 10 zeigt den Ablauf der Posenberechnung, wobei die einzelnen Zwischenschritte, die dabei von OpenCV generiert werden in Rot dargestellt werden. Für den Ablauf wird vorausgesetzt, dass zuvor die Marker im aktuellen Frame entweder automatisiert oder durch den Nutzer erkannt wurden.

Zur besseren Visualisierung und zum optischen Erkennen von grob falschen Posen zur Laufzeit werden im Anschluss an das Lösen des PnP-Problems das Weltkoordinatensystem in seinem Ursprung und die Markerpositionen in den aktuellen Frame projiziert.

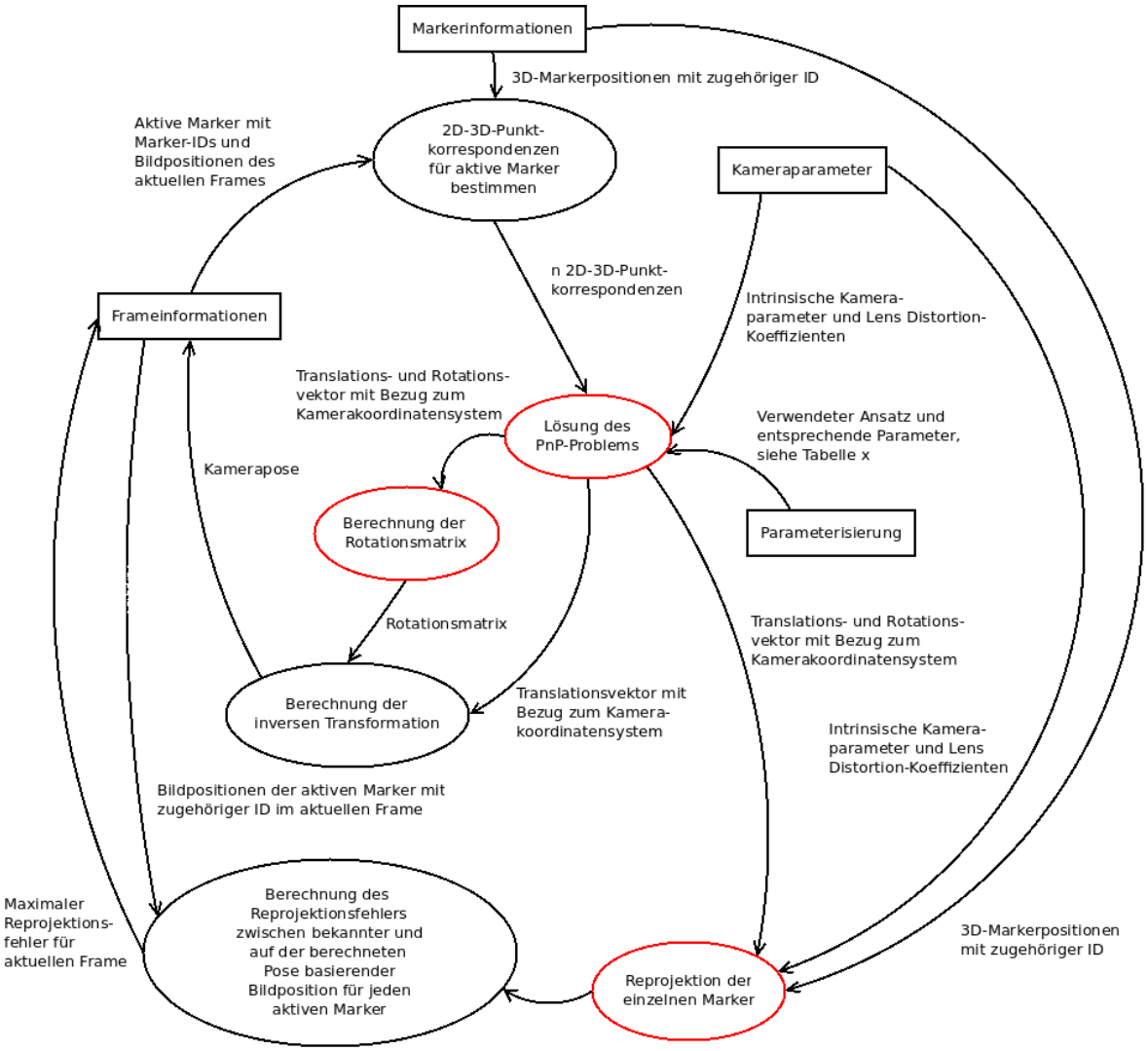


Abbildung 10: Schritte der Posenberechnung und Berechnung des maximalen Reprojektionsfehlers, nachdem die Markerpositionen im aktuellen Frame bestimmt wurden

5.5 Weitere Funktionalität

Neben den zur Posenberechnung benötigten Programmteilen sind eine Reihe von weiteren Funktionen implementiert. Standardmäßig werden dabei die jeweiligen Auswahlmöglichkeiten in der linken oberen Ecke des Bildschirms angezeigt und je nach aktueller Auswahl angepasst. Abb. 11 zeigt die Wahlmöglichkeiten nach Start des Programms.

Am wichtigsten ist dabei die Möglichkeit, die berechneten Posen für eine mögliche Weiterverarbeitung abzuspeichern. Dazu werden bei der entsprechenden Eingabe durch den Nutzer alle Frames, für die die Pose berechnet wurde, in eine CSV-Datei geschrieben. Hinterlegt wird dabei neben der Framenummer die Rotationsmatrix und der Translationsvektor sowie der maximale Reprojektionsfehler im jeweiligen Frame. Die Verwendung des CSV-Formats erlaubt die einfache Importierung der Daten in Programmen zur Weiterverarbeitung wie z. B. Microsoft Excel oder GNU Octave. Letzters wurde im Rahmen dieser Arbeit für die Evaluierung genutzt.

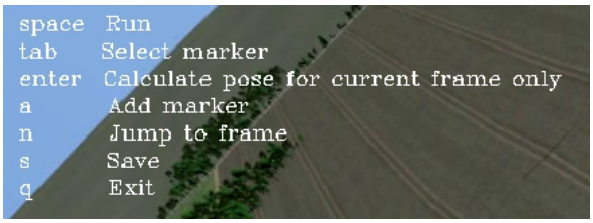


Abbildung 11: Standardmenü beim Programmstart

Daneben ist es möglich, Marker nachträglich zu editieren oder zu entfernen, nachdem der entsprechende Marker ausgewählt wurde. Dazu wird der aktuell ausgewählte Marker farbig umrandet, um ihn hervorzuheben, und kann dann bearbeitet werden.

Außerdem kann zu beliebigen Frames gesprungen werden, was inbesondere dann interessant ist, wenn die Pose nur für ausgewählte Frames als Referenzwert benötigt wird. Wird dabei zu einem Frame gesprungen, in dem die Kamerapose bereits berechnet wurde, so wird die zur Berechnung genutzte Markerauswahl automatisch beibehalten und angezeigt. Es ist dann beliebig möglich, die Marker zu ändern und die Berechnung erneut zu starten. Bei einer automatisierten Berechnung werden bei allen nachfolgenden Frames die zuletzt angewählten Marker ebenfalls übernommen, sodass kein mühsames Editieren der Markerauswahl in einzelnen Frames nötig ist.

Zur Fehlerausgabe werden Statusnachrichten im unteren linken Bildrand eingeblendet, die für eine variable Zeit eingeblendet werden oder alternativ über einen beliebigen Tastendruck ausgeblendet werden können.

Die zuvor erwähnte Konfigurationsdatei dient der Initialisierung des Programms zur Posenberechnung. Aktuell können dort die in Tabelle 1 gezeigten Werte hinterlegt werden. Die Konfiguration kann jedoch beliebig erweitert werden.

Parameter	Standardwert
Intrinsische Kameraparameter	
Lens Distortion-Koeffizienten	
PnP-Algorithmus	Iterativ (RANSAC)
Maximaler Reprojektionsfehler	5 Px
Größe der Suchmaske für Markerfindung	15 Px
Markeranzahl	10
Markerpositionen	

Tabelle 1: Parameter der Konfigurationsdatei mit Standardwert, falls vorhanden

6 Simulation

Aufgrund der Tatsache, dass es sich bei dieser Arbeit um eine Vorstudie handelt, existieren noch keine realen 2D-Luftaufnahmen, bei denen nötige Marker für eine Posenbestimmung auf dem Boden platziert wurden. Deshalb liegt ein wichtiger Teil dieser Arbeit in der Erstellung einer Simulation der Flugphase des Kites, die für Tests der in Kapitel 5 vorgestellten Software geeignet ist. Die Simulation orientiert sich dabei an realen Aufnahmen einer am Kite montierten GoPro-Kamera, die einen Pumping Cycle des Kites ohne am Boden platzierte Marker zeigt. Abb. 12 zeigt dafür einen Frame des realen Videos sowie der Simulation.

(a)

(b)

Abbildung 12: Übersicht (a) einer Aufnahme der am Kite montierten Kamera und (b) einem gerenderten Frame der erstellten Simulation

6.1 Umsetzung

Für die Simulation kommt die freie 3D-Grafiksoftware *Blender* zum Einsatz, die bereits seit 1994 kontinuierlich weiterentwickelt wird. *Blender* bietet die Möglichkeit, dreidimensionale Modelle zu modellieren und simulieren, und integriert darüber hinaus die Skriptsprache *Python*.

Da eine detaillierte Erläuterung aller Einzelheiten der Simulation sehr aufwendig ist und im Detail dem Inhalt dieser Arbeit keinen Mehrwert bietet, beschränkt sich die Beschreibung der Umsetzung auf die für die Bildverarbeitung relevanten Aspekte.

Die Basis der Simulation bietet eine ca. 9 km^2 große quadratische Grundfläche. Diese Größe ist deshalb erforderlich, weil die Sicht des Kites in größeren Höhen insbesondere während einer Drehung sehr weitläufig ist.

Die Grundfläche ist weiter in quadratische Flächen mit einer Größe von 50 m^2 eingeteilt. Die so geteilte Fläche wird anschließend verformt, um entfernte Hügel und Täler bzw. grobe Unebenheiten zu simulieren (Abb. 13). So werden grobe, aus großen Höhen erkennbare Unebenheiten erzeugt, die jede reale Luftaufnahme in Abhängigkeit von der Landschaft prägen.

Im Fokus steht insbesondere der Bereich, in dem später die Marker platziert werden sollen, da hier die Marker erkannt werden müssen. Dieser in relativer Nähe zur Bodenstation liegende Bereich sollte deshalb auch in der Simulation realistischer dargestellt werden. Aus diesem Grund ist diese rund 4000 m^2 große Fläche in der Simulation feiner in 4.5 x 4.5 m große Quadrate unterteilt, die es erlauben, detailliertere Unebenheiten des Bodens zu simulieren (Abb. 13). Dass dabei nicht die gesamte Grundfläche so detailliert dargestellt wird, liegt daran, dass sonst die Rechenzeit deutlich erhöht werden würde. Dabei wäre eine detaillreichere Darstellung außerhalb des Bereichs der Marker aus den Höhen des Kites ohnehin nicht zu erkennen. Sie wird deshalb bewusst nicht feiner simuliert.

Ein wesentlicher Bestandteil der Simulation ist die simulierte Kamera, die in *Blender* in Form eines Kameraobjekts dargestellt wird. Das Kameraobjekt verfügt im Gegensatz zu anderen Objekten neben den allgemeinen Parametern zur Positionierung und Ausrichtung über viele Einstellungen, die reale Kameras so genau wie möglich simulieren sollen. Diese Einstellungen beeinflussen daher maßgeblich die gerenderte Simulation. Neben diesen Kameraparametern, zu denen unter anderem die Brennweite und Größe des Kamerasensors gehören, ist es beim Rendern der Simulation von hoher Bedeutung, das sogenannte Antialiasing möglichst gut durchzuführen. Antialising dient dazu, die Kanten in einem gerenderten Bild zu glätten. Dazu wird zur Bestimmung des Wertes eines Pixels nicht nur der Pixel selber untersucht, sondern je nach Algorithmus auch Bildpunkte in der Nachbarschaft des aktuellen Pixels. Konkret kommt für die Simula-

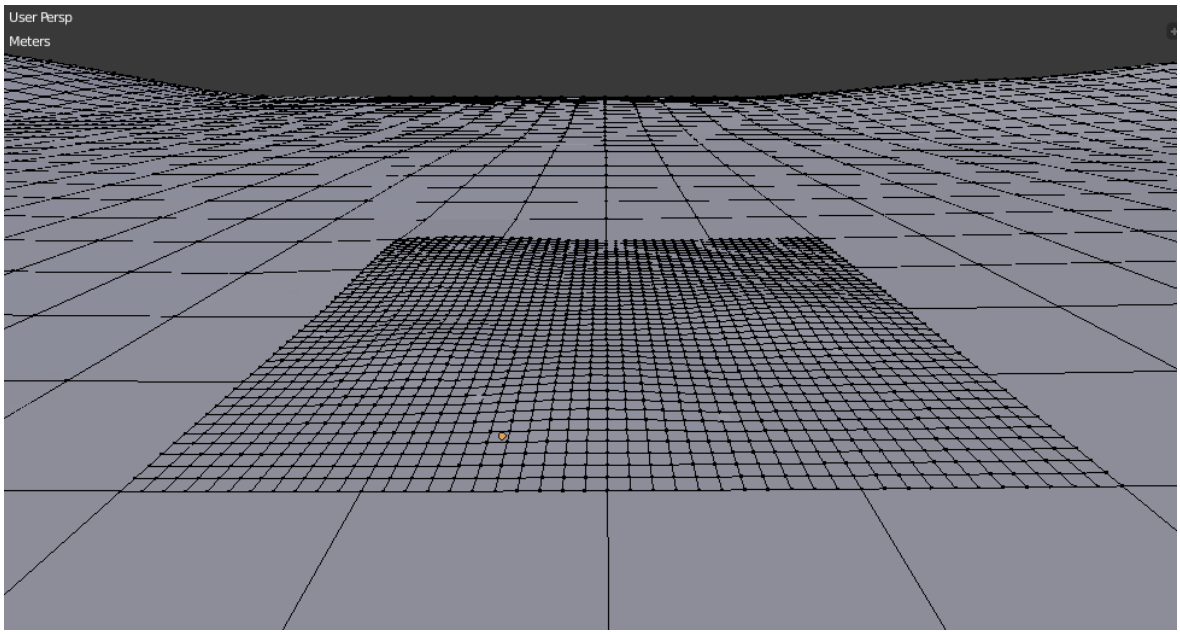


Abbildung 13: Grobe und feine Verformungen der quadratischen Teilflächen zur Simulation von Unebenheiten des Untergrunds

tion ein Mitchell-Netravali-Filter zur Anwendung, der bereits in *Blender* integriert ist. Antialiasing ist wesentlicher Bestandteil beim Rendern der Simulation, da ohne eine Glättung zufällige Kanten im gerenderten Bild entstehen könnten, die bei der Posenberechnung zu starkem Abweichungen führen und keine realistischen Kameraaufnahmen widerspiegeln würden.

Ein weiterer relevanter Aspekt der verwendeten Kamera, der nicht direkt im Kameraobjekt konfiguriert werden kann, ist die Linsenverzeichnung (engl. Lens Distortion). Um die Linsenverzeichnung realistisch darzustellen, wurde eine GoPro-Kamera zunächst kalibriert und anschließend einige Testaufnahmen aufgenommen. Anhand dieser Aufnahmen wurde dann die Linsenverzeichnung der verwendeten Kamera mithilfe von OpenCV annähernd bestimmt und das Ergebnis auf die Berechnung in der Simulation übertragen. In ersten Tests hat sich jedoch gezeigt, dass die künstliche Lens Distortion nicht den Effekten einer realen Kamera entspricht (siehe Kapitel 7).

Die Bewegung des Kameraobjekts wird dabei so kontrolliert, dass es in den relevanten Höhen von ca. 130 - 280 m einen kompletten Pumping Cycle bestehend aus Energieerzugungsphase, Transferphase und schließlich Rückholphase durchläuft. Abb. 14 zeigt die auf den Daten der einzelnen Frames basierende Trajektorie der Simulation. Dabei entfernt sich der Kite zunächst mit einer Geschwindigkeit von knapp 0,5 m/s Seillänge von der Basisstation, ehe er in die Transferphase übergeht und schließlich zurückgeholt wird. Dazu wird eine liegende Acht in *Blender* als geschlossene 2D-Kurve zur Darstellung der Energieerzeugungsphase sowie ein 2D-Pfad in Richtung horizontaler Richtung zur Basisstation genutzt, der die Transferphase simuliert. Das Kameraobjekt wird abhängig von der Höhe an den jeweiligen Pfad gebunden. Dies wird über sogenannte

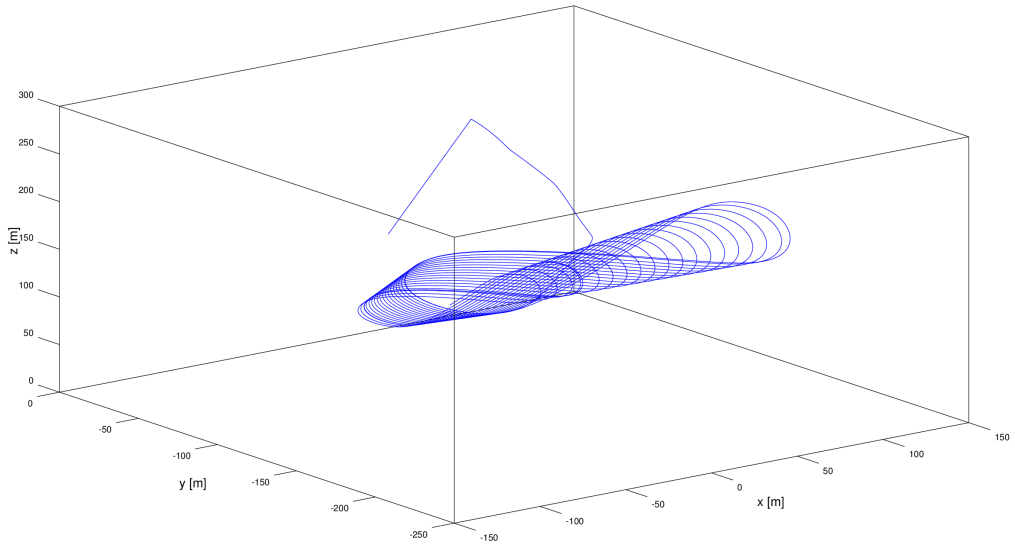


Abbildung 14: Trajektorie des Kites basierend auf den Positionsdaten der Simulation über 10900 Frames beginnend bei einer Seillänge von 130 m in der Reel-Out-Phase. Wird eine Seillänge von ca. 273 m erreicht, wird zunächst die Transferphase und abschließend die Reel-In-Phase durchlaufen.

Blender Constraints realisiert, bei denen es sich um Bedingungen handelt, mit denen die Position, Rotation und Skalierung eines Objekts zu einem bestimmten Zeitpunkt beeinflusst werden kann. Abb. 15 zeigt eine Übersicht über die Flugphase in der Simulation, in der das Kameraobjekt von den erwähnten Constraints kontrolliert wird. Dabei können Parameter wie Geschwindigkeit, Flughöhe und Flugmuster bei Bedarf weiter angepasst werden.

Vor allem bei größeren Entfernungen des Kites vom Boden wird eine weitläufige Landschaft erfasst, in der in der Realität viele Waldstücke, Felder, Wege und andere Objekte sowie deren Schatten sichtbar sind. Diese Objekte stellen mögliche Fehlerquellen bei der Markerdetektion dar, falls sie fälschlicherweise als Eckpunkte von Markern erkannt werden. Daher werden diese Objekte auch in der Simulation dargestellt (siehe Abb. 12 b). Dabei kommen für die Waldstücke sogenannte Partikelsysteme zum Einsatz. Partikelsysteme bieten die Möglichkeit, Objekte als Emitter von Partikeln zu verwenden. In diesem Fall wurden einige detaillarme Baummodelle erstellt, die dann von den Waldstücken in zufälliger Verteilung und mit definierter Anzahl als Partikel emittiert werden.

Anschließend werden Texturen in Form von UV-Maps eingefügt, die den verschiedenen Objekten eine realistische Textur verleihen. Bei UV-Maps handelt es sich in Blender um 2D-Muster, die nach den gewünschten Vorgaben beim Rendern auf die 3D-Objekte projiziert werden.

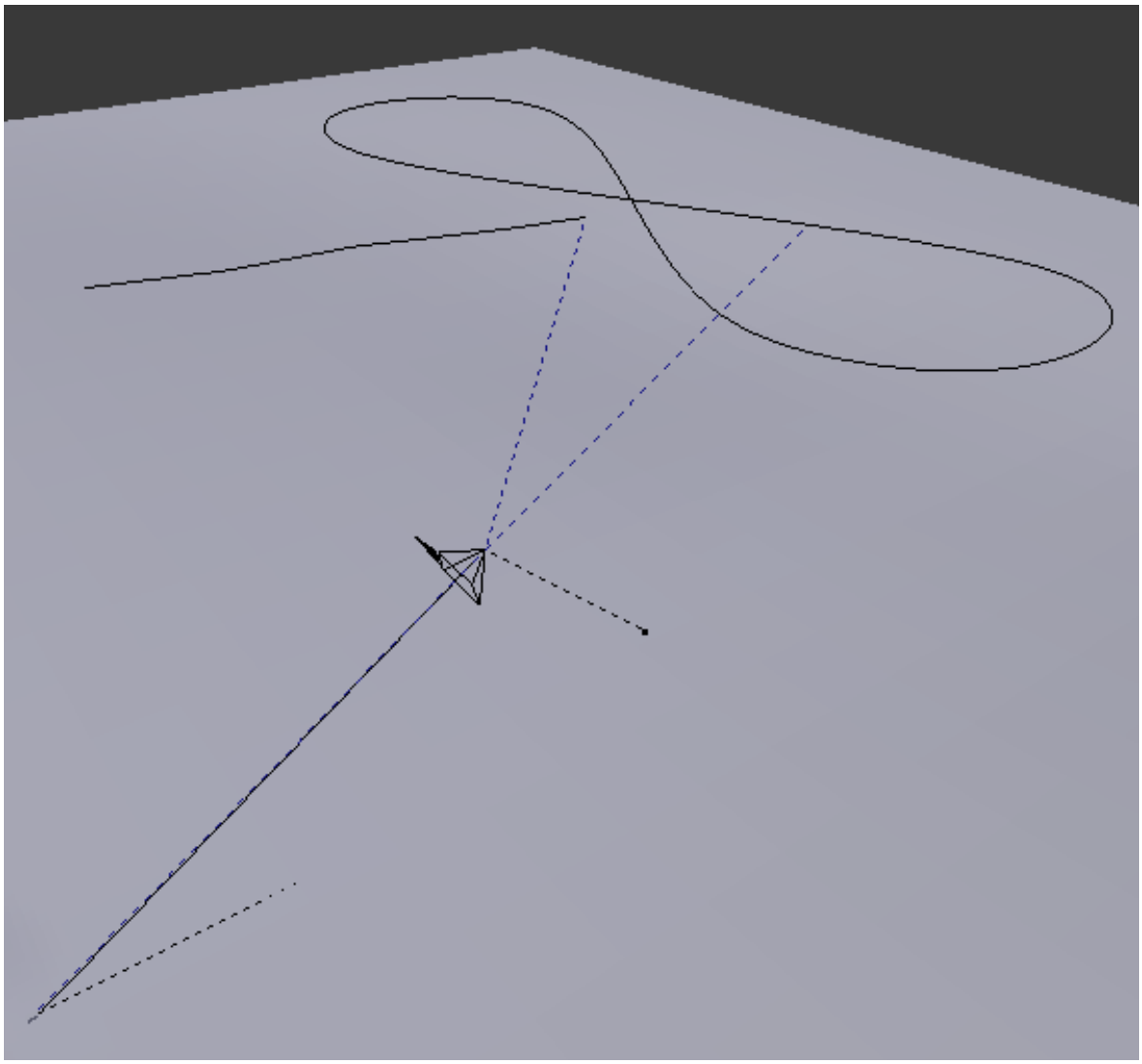


Abbildung 15: Übersicht der Umsetzung des Pumping Cycles mit Grundfläche (hellgrau), Kameraobjekt mit Seil, flacher Acht für Reel-Out-Phase, Pfad für Transferphase sowie als gestrichelte Linien dargestellte Constraints

Die zu Beginn dieses Kapitels angesprochene Python-Unterstützung erlaubt die Verwendung von Skripten, um die Simulation zu erweitern. Konkret wird das im Rahmen dieser Arbeit genutzt, um absolute Werte der 3D-Position der Marker zu gewinnen, die bei Tests des Programms als Richtwerte (engl. *Ground Truth*) genutzt werden können. Neben den Markerpositionen ist die Pose der Kamera in Bezug auf das Weltkoordinatensystem als Ground Truth von großer Bedeutung, weshalb die entsprechende Transformationsmatrix bzw. die Rotationsmatrix und der Translationsvektor ebenfalls als Ground Truth gespeichert werden. Dazu wurde ein Python-Skript entwickelt, das je nach Nutzereingabe die gesamte Simulation oder bestimmte Abschnitte rendert und dabei für jeden Frame die entsprechenden Daten erzeugt. Die Daten werden dabei automatisch in einer Log-Datei gespeichert, die dann direkt als Ground Truth bei späteren Tests genutzt werden können. Das Skript befindet sich im digitalen Anhang dieser Arbeit.

6.2 Eigenschaften und Einschränkungen

Das Ziel der Simulation ist die möglichst realistische Darstellung der für die Markerdetektion bei realen Luftaufnahmen ausschlaggebenden Einflüsse. Im Folgenden sollen die simulierten Einflüsse dargestellt werden und Grenzen der Simulation aufgezeigt werden.

Die Grundvoraussetzung für eine Posenbestimmung basierend auf der Simulation ist zum einen die maßstabsgetreue Darstellung der Umgebung und zum anderen die an die reale Kamera angelehnte Parameterisierung des Kameraobjekts. Beide Voraussetzungen werden, wie in Kapitel 6.1 beschrieben, erfüllt. Zusätzlich zur Parameterisierung des Kameraobjekts wird die Linsenverzerrung der Kamera in der Simulation berücksichtigt. Dies wird in *Blender* im Anschluss an das Rendern eines Frames über sogenannte Composite Nodes realisiert. Dazu wird nachträglich für das gerenderte Bild eine definierte Linsenverzerrung berechnet und über das Bild gelegt.

Daneben sind die Marker über sogenannte Modifier an die unterliegende Oberfläche angepasst. Sie liegen also nicht in einer perfekten Ebene, sondern sind abhängig von ihrer Position auf dem simulierten Untergrund leicht gewölbt. Darüber hinaus sind sie minimal verzerrt gerendert, sodass auch die einzelnen schwarzen und weißen Flächen nicht bei allen Markern perfekte Quadrate erzeugen, um so mögliche Fehler, die z. B. bei der Herstellung der Marker entstehen können, darzustellen. Je nach Material, das später für die Marker verwendet werden soll, muss außerdem mit einer Lichtreflektion durch die Markeroberfläche gerechnet werden. Deshalb ist die Reflektion der simulierten Marker variabel implementiert und reflektiert je nach Winkel und Unebenheit des Untergrunds einfallendes Licht der Simulation mit einer definierten Intensität.

Im Video der realen Flugphase des Kite-Prototypen von SkySails ist außerdem zu erkennen, dass die Kamera während des Fluges leicht vibriert. Aus diesem Grund wird für jede der drei Achsen des lokalen Koordinatensystems des Kameraobjekts eine zufällige Störung des Rotationswinkels gelegt, die sowohl zeitlich als auch in ihrer Intensität unabhängig voneinander sind. So entsteht eine scheinbare Vibration der simulierten Kamera, die im zeitlichen Verlauf innerhalb definierter Grenzen variiert.

Demgegenüber stehen jedoch einige Faktoren, die nicht oder nur unzureichend durch eine Simulation dargestellt werden können. So kann es bei einer realen Kamera zu rapiden Wechseln des Lichteinfalls innerhalb angrenzender Frames kommen. Obwohl die Simulation Lichteffekte berücksichtigt, können diese nicht so abrupt dargestellt werden, wie sie ggf. in realen Aufnahmen auftreten könnten. Außerdem ist die angenommene Flugphase in Form einer flachen Acht in der Simulation ideal. In realen Aufnahmen weicht diese jedoch zumeist ab, was zu unbekannten Einflüssen auf die Posenberechnung (z. B. direkte Sonneneinstrahlung) führen kann, die die Simulation nicht erfasst.

Außerdem kommt bei der verwendeten Kamera ein Rolling Shutter zum Einsatz, sodass die Pixelreihen eines Frames zeitlich geringfügig versetzt aufgenommen wurden. Dieses Verhalten ist in der Simulation nicht abgebildet.

Deshalb ist es unerlässlich, neben der Simulation ein Modell für Testaufnahmen unter Verwendung der realen Kamera durchzuführen (siehe Kapitel 7).

7 Evaluation

Im Folgenden wird die Berechnung der Pose in ihren Elementen getestet, um Aussagen über die Qualität der Posenberechnung zu erhalten sowie mögliche Fehlerfaktoren zu erkennen und später in der Anwendung zu berücksichtigen.

Voraussetzung für alle Tests ist die Verwendung einer kalibrierten Kamera. Das gilt sowohl für die Simulation in Form einer virtuellen Kalibrierung als auch für die Modellversuche mit realer Kamera. Dabei wird für die Simulation ein animiertes Kalibrierobjekt in Form eines Schachbrettmusters mit 20 x 12 Kacheln genutzt, das ebenfalls in *Blender* im Rahmen dieser Arbeit erzeugt wurde, und für die reale Kamera ein Schachbrettmuster mit 20 x 15 Kacheln (siehe Abb. 16). Da im Gegensatz zur realen Kamera bei einer normalen Blenderanimation keine Lens Distortion-Effekte auftreten, sollte zunächst ein Wert für die künstliche, nachträglich in *Blender* erzeugte, Lens Distortion bestimmt werden. Dies geschieht experimentell, indem eine reale Aufnahme des Kalibrierobjekts mit der GoPro-Kamera in *Blender* dimensionsgetreu nachgestellt wird.

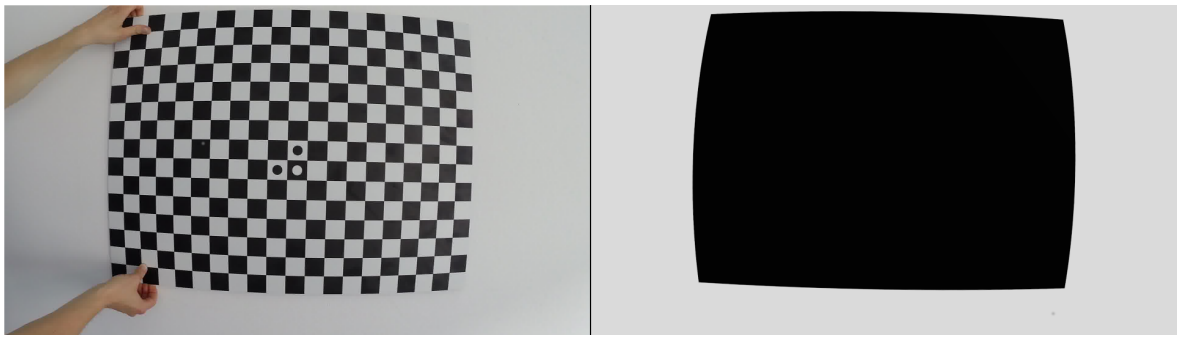


Abbildung 16: Links eine Aufnahme mit der verwendeten GoPro-Kamera, rechts die dimensionsgetreue Nachstellung mit dem experimentell bestimmten Wert des Distortion-Parameters in Blender von 0.14

Abb. 16 zeigt den Vergleich zwischen realer Aufnahme und experimentell bestimmter Lens Distortion in *Blender*. Es ist zu erkennen, dass, im Gegensatz zur GoPro, die künstliche Distortion in *Blender* in horizontaler Richtung ausgeprägter ist als in der vertikalen. Bei dem Standardverfahren zur Erzeugung künstlicher Lens Distortion in *Blender*, das in Abb. 16 verwendet wurde, lässt sich die Distortion nur über einen Parameter und somit nicht in einzelne Richtungen konfigurieren. In der aktuellen Blenderversion ist es jedoch möglich, anhand einer realen Kameraaufnahme die intrinsischen Kameraparameter (siehe Kapitel 4.1) der verwendeten Kamera grafisch zu ermitteln, um so eine realistische Lens Distortion zu erzeugen, die auf dem gleichen Kameramodell wie OpenCV basiert. Allerdings stand die benötigte Blenderversion zum Zeitpunkt der Erstellung dieser Arbeit auf den universitären Rechnern, die zum Rendern der Animation verwendet wurden, nicht zur Verfügung und konnte dementsprechend nicht verwendet werden. Aus diesem Grund und weil das simplere Blendermodell, das der

Berechnung der Distortion in Abb. 16 zugrunde liegt augenscheinlich nicht dem einer realen Kamera entspricht, wurde darauf verzichtet, bei der Simulation eine künstliche Lens Distortion zu verwenden. Dies ist zulässig, weil bei Kenntnis der intrinsischen Kameraparameter die Effekte der Lens Distortion theoretisch vollständig berechnet und die Bilder ohne Distortion und somit wie in der Simulation dargestellt werden können.

Für die am Modell durchgeführten Tests wird hingegen eine GoPro Hero3+ Silver Edition verwendet, bei der eine tonnenförmige Lens Distortion auftritt. Dabei wird die Einstellung für das mittlere Field of View (FOV) der Kamera bei einer Auflösung von 1920 x 1080 Px für die Tests verwendet. Für die Modellaufnahmen wird die Kamera auf ein Dreibeinstativ montiert und für jede Messung wird zunächst von Hand die Kameraposition als Referenzwert bestimmt. Dabei befindet sich das optische Zentrum der Kamera, auf das sich die berechnete Position der Kamera bezieht, im ersten Drittel des Objektivs. Der Fehler, der bei der manuellen Bestimmung der Referenzwerte nicht zu vermeiden ist, liegt bei ca. 1 cm. Eine Bestimmung der Orientierung von Hand als Referenzwert ist hingegen kaum möglich, sodass sich die Modelltests auf eine Evaluation des Positionsfehlers beschränken. Dies ist zulässig, da bei einer inkorrekten Bestimmung der Orientierung bei der Lösung des PnP-Problems auch keine korrekt Positionsbestimmung stattfinden könnte. Deshalb wird davon ausgegangen, dass bei korrekten Positionsdaten auch die Orientierung zumindest näherungsweise korrekt ist. Trotzdem ist eine Untersuchung der Orientierung empfehlenswert und wird anhand der Simulation durchgeführt.

Dabei ist außerdem zu beachten, dass für das Modell umgesetzt wird, dass alle Marker in einer Ebene liegen, um die Auslegung der Marker zu vereinfachen. Für das Modell wurden dazu Marker erstellt, die bei einer Größe von ca. 2,8 x 2,8 cm aus demselben Stoff bestehen, der auch später bei den realen Markern verwendet werden soll. So können bereits erste Tests des Materials durchgeführt werden.

Für die Tests wurde bei der Kalibrierung der GoPro-Kamera ein mittlerer Reprojektionsfehler (siehe Kapitel 4.1) $QMW_{real} = 0,71\ Px$ und für die Simulation $QMW_{sim} = 0,02\ Px$ erreicht. Die sehr gute Kalibrierung anhand der Simulation ist insbesondere auf die fehlende Lens Distortion zurückzuführen. Bei beiden Kalibrierungen wurde die OpenCV-Standardmethode verwendet, die einige Einschränkungen beinhaltet. So ist es beispielsweise erforderlich, dass immer alle inneren Kanten des Kalibriermusters im Bild zu erkennen sind. Dies erschwert die Kalibrierung der Randbereiche enorm. $QMW_{real} = 0,71\ Px$ ist für die Evaluation der Posenberechnung zwar ausreichend genau, allerdings könnte hier durch andere Kalibrierverfahren die Kalibrierung verbessert bzw. die Genauigkeit der Posenberechnung erhöht werden.

Falls nicht anders angegeben, wird der maximal zulässige Reprojektionsfehler zwischen ausgewählter und reprojizierter Markerposition für die einzelnen Marker auf 5 Px in der Simulation bzw. 10 Px für Modelltests festgelegt.

7.1 Posenberechnung

Im Vordergrund der Evaluation steht die Bestimmung der Pose, die das Endergebnis dieser Arbeit darstellt. Dabei werden zumeist der Orientierungs- und Positionsfehler dargestellt und bei Bedarf der Reprojektionsfehler und weitere Besonderheiten bei den einzelnen Tests erläutert.

Als Fehlermaß für die Position wird dazu der euklidische Abstand d zwischen den Positionsvektoren der Referenzposition (Ground Truth) pos_{gt} und der ermittelten Position der Kamera pos_{est} verwendet:

$$d(pos_{gt}, pos_{est}) = \|pos_{gt} - pos_{est}\|$$

Als Fehlermaß für die Orientierung wird die Axis Angle-Repräsentation gewählt, die wie folgt definiert ist ([Huy09], S. 159):

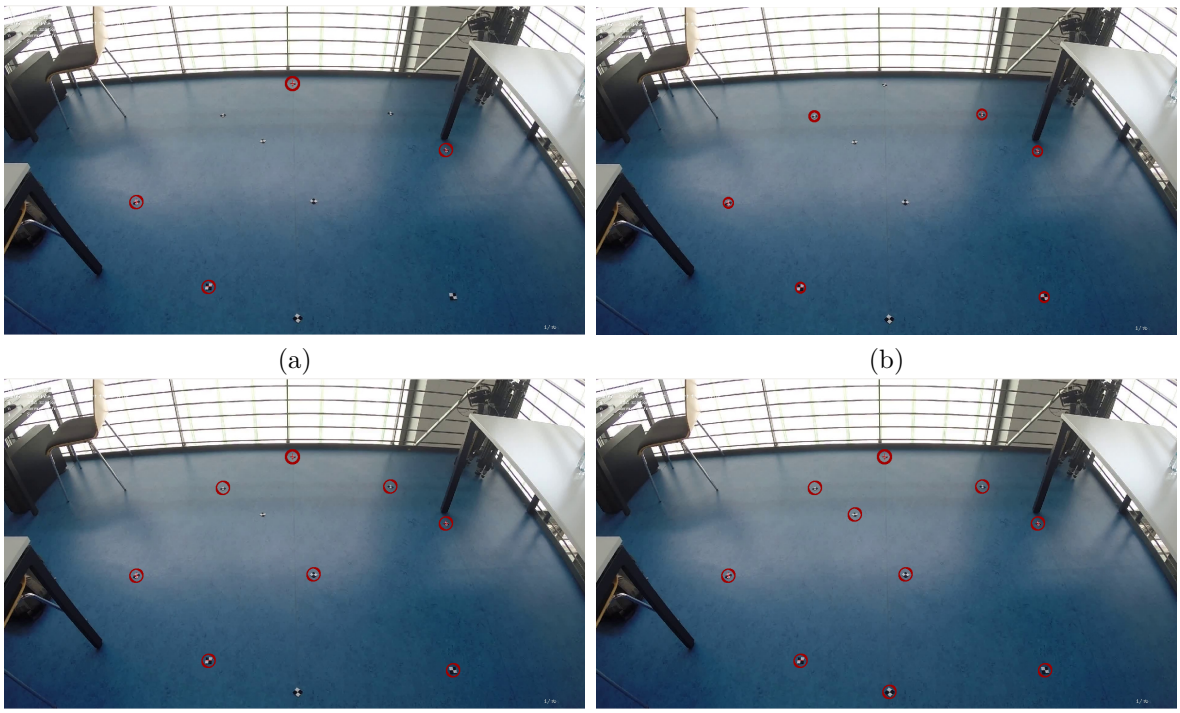
$$\phi(R_{gt}, R_{est}) = \|log(R_{gt}R_{est}^T)\|$$

R_{gt} sowie R_{est} sind die Rotationsmatrizen der aktuellen Berechnung für die Ground Truth-Werte sowie die ermittelte Pose.

Zur anschaulichen Darstellung wird in einigen Fällen der Fehler in Abhängigkeit von der Seillänge bzw. Entfernung des Kites von der Ground Station abgebildet. Dabei ist zu beachten, dass der als Seillänge definierte Abstand nicht der tatsächlichen Seillänge entspricht, sondern vielmehr der Entfernung vom Ursprung des Koordinatensystems bzw. der Ground Station des Kites zum optischen Zentrum der Kamera. Zur Veranschaulichung wird dieser Abstand jedoch als Seillänge bezeichnet.

Zunächst wird untersucht, wie sich die verschiedenen Algorithmen zur Posenbestimmung bei einer realen Aufnahme anhand des Modells und einer nahezu idealen Kalibrierung bei der Simulation verhalten. Abb. 17 zeigt den zugehörigen Aufbau des Modells bei einer Ausgangshöhe der Kamera von 143 cm. Dabei wird die Kamerahöhe schrittweise variiert, bis eine Höhe von 191 cm erreicht wird. Die Höhe wird dabei zum einen durch die Raumhöhe und zum anderen durch das Stativ begrenzt. Bei einer größeren Höhe müssten außerdem die Beine des Stativs ausgefahren werden, was die Referenzmessung der Position zusätzlich verfälschen würde.

Zusätzlich zur Variation der Kamerahöhe wird die Markeranzahl für die einzelnen Algorithmen varriert. Die jeweils aktiven Marker sind ebenfalls in Abb. 17 dargestellt.



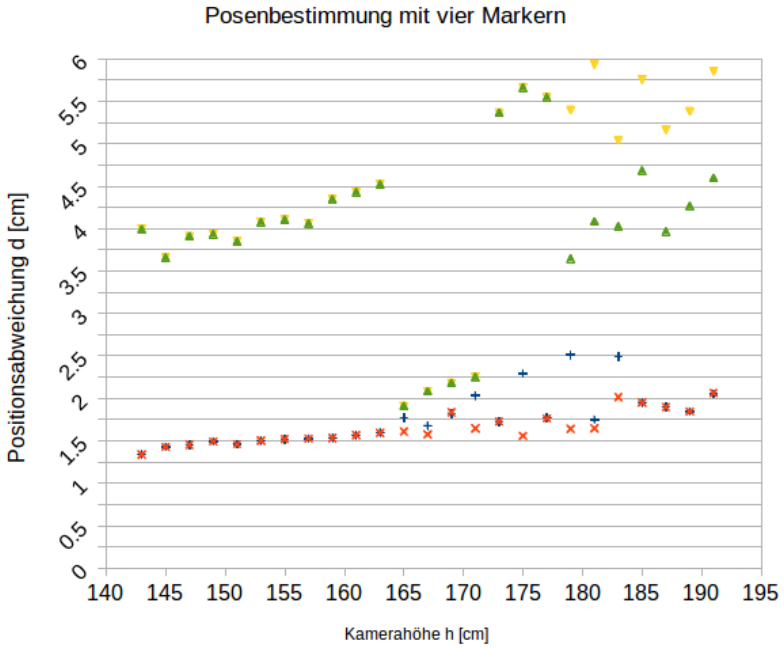
(a) (b)

(c) (d)

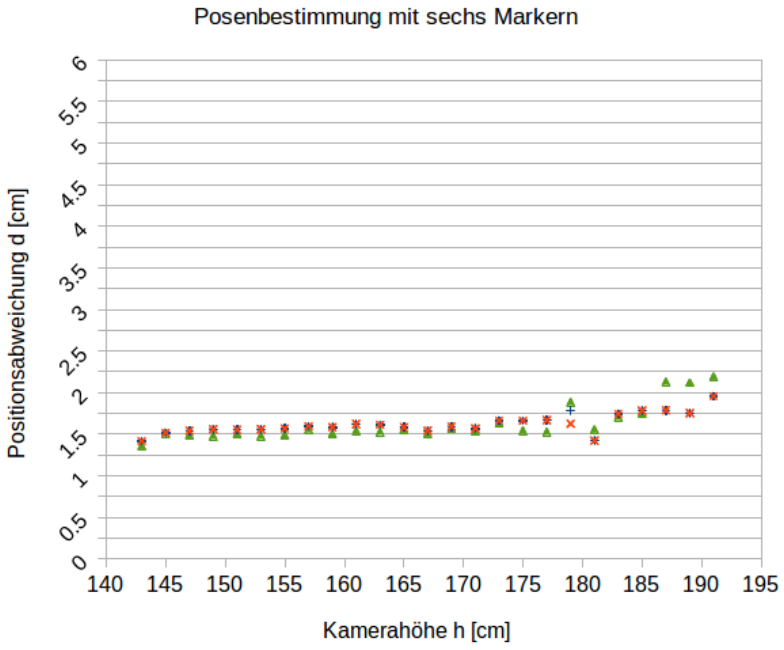
Abbildung 17: Modellaufnahmen bei der gezeigten Ausgangshöhe der Kamera von 143 cm für a) vier, b) sechs, c) acht, d) zehn Marker. Die Höhe der Kamera wird für die Tests von 143 auf 191 cm in 2 cm Schritten variiert. Die Orientierung der Kamera bleibt unverändert.

Überschreitet die Reprojektion der Bildposition eines Markers diesen Wert, so wird die Berechnung der Pose als gescheitert angesehen und kein Wert dargestellt.

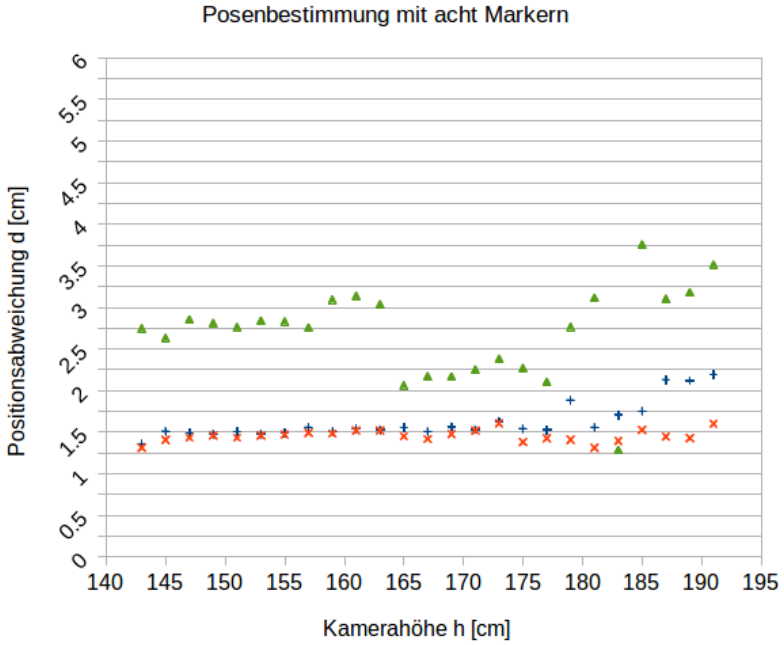
Abb. 18 zeigt die Positionsabweichungen der jeweils berechneten Position zur zuvor ermittelten Referenzposition für die einzelnen Algorithmen für die jeweils in Abb. 17 dargestellte Markerauswahl. Dabei ist zu erkennen, dass der P3P-Algorithmus empfindlich davon abhängt, welche Marker zur Posenbestimmung verwendet werden. Demgegenüber liefert die iterative Lösung sowohl mit als auch ohne RANSAC-Anwendung deutlich stabilere Ergebnisse, die in den getesteten Höhen einen maximalen Positionsfehler von unter 2,2 cm aufweisen. Inbesondere die iterative Lösung unter Verwendung des RANSAC-Algorithmus liefert bei einer größeren Anzahl an Markern bessere Ergebnisse, die bei zehn Markern den maximalen Positionsfehler in einer Höhe von 191 cm auf unter 1,72 cm senken. Dabei ist wie zu erwarten ein Abfall der Genauigkeit bei steigenden Höhen zu erkennen. Tabelle 2 zeigt die absolute mittlere Änderung des Positionsfehlers pro Zentimeter Höhe für die Algorithmen, die einen näherungsweise linearen Fehlerverlauf mit steigender Höhe aufweisen. Dabei ist zu beachten, dass der Ansatz von Moreno (EPnP) für alle Testfälle ungeeignete Resultate lieferte, bei denen der Reprojektionsfehler der Marker weit über 10 Px lag.


Posenbestimmung mit vier Markern
Positionsabweichung d [cm]
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
5
5,5
6
140
145
150
155
160
165
170
175
180
185
190
195
Kamerahöhe h [cm]

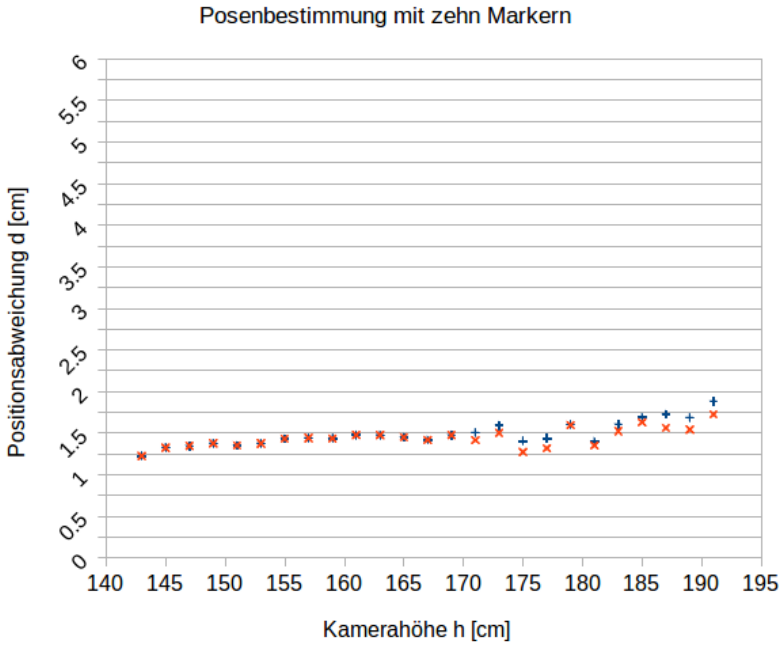
(a)


Posenbestimmung mit sechs Markern
Positionsabweichung d [cm]
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
5
5,5
6
140
145
150
155
160
165
170
175
180
185
190
195
Kamerahöhe h [cm]

(b)



(c)



(d)

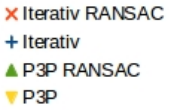


Abbildung 18: Positionsfehler für die verschiedenen Algorithmen mit (a) vier, (b) sechs, (c) acht und (d) zehn Markern für die in Abb. 17 dargestellten Konstellationen

Algorithmus	Markeranzahl			
	4	6	8	10
Iterativ		0,006	0,012	0,0085
Iterativ RANSAC	0,012	0,0058	0,001	0,005
P3P				
P3P RANSAC		0,012		
EPnP				
EPnP RANSAC				

Tabelle 2: Absoluter Anstieg des Positionsfehlers in cm pro cm Höhe für eine variierende Markeranzahl basierend auf linearer Regression

Aufgrund der Stabilität der Ergebnisse des iterativen Ansatzes, wird dieser für die weiteren Modelltests verwendet und standardmäßig beim Start des Programms genutzt. Um auf Ausreißer besser reagieren zu können, kommt dabei nicht allein der reguläre iterative Ansatz, sondern in Kombination mit dem RANSAC-Algorithmus zum Einsatz. Dabei sei darauf hingewiesen, dass der iterative Ansatz insbesondere in Kombination mit dem RANSAC-Algorithmus abhängig von den maximal zulässigen Iterationen deutlich langsamer ist als beispielsweise der EPnP-Algorithmus. Echtzeitfähigkeit ist jedoch keine Anforderung an das Programm. Deshalb wird die längere Laufzeit zugunsten der Stabilität in Kauf genommen. Aus demselben Grund werden auch keine Tests zur Laufzeit der einzelnen Algorithmen durchgeführt. Sollte in Zukunft Rechenzeit ein relevanter Aspekt werden, bietet es sich an, Laufzeittests der PnP-Algorithmen durchzuführen und die Parameterisierung der iterativen RANSAC-Methode anzupassen oder zu ersetzen.

Wird von einem weiterhin linearen Verlauf ausgegangen, so ist für eine Höhe von 300 cm für den iterativen Ansatz unter Verwendung von RANSAC mit allen verfügbaren Markern ein Positionsfehler basierend auf einer linearen Regression von ca. 2,15 cm zu erwarten.

Um die Algorithmen anhand der Simulation zu testen, wird eine Position innerhalb der flachen Acht während der Reel-Out-Phase gewählt und diese für Höhen von 129 – 272 m simuliert und ausgewertet. Dazu wird erneut die Markeranzahl variiert. Allerdings werden lediglich vier und sechs Marker berücksichtigt, da in den tieferen Höhen keine weiteren Marker im Bild sichtbar sind.

Dabei wird in Abb. 20 deutlich, dass der EPnP-Algorithmus durchaus genaue Messergebnisse liefern kann: Bei der Auswahl von sechs Markern wird ein nahezu konstanter Orientierungsfehler von 0,05° unabhängig von der aktuellen Höhe erreicht. Dabei erhöht sich der Positionsfehler mit ca. 0,0023 m/m Seillänge. Dies ist von allen Algorithmen das beste Ergebnis. Allerdings ist es bei Tests immer wieder vorgekommen, dass bestimmte Markerkonstellationen völlig unbrauchbare Ergebnisse und Reprojektionsfehler von über 1000 Px lieferten.

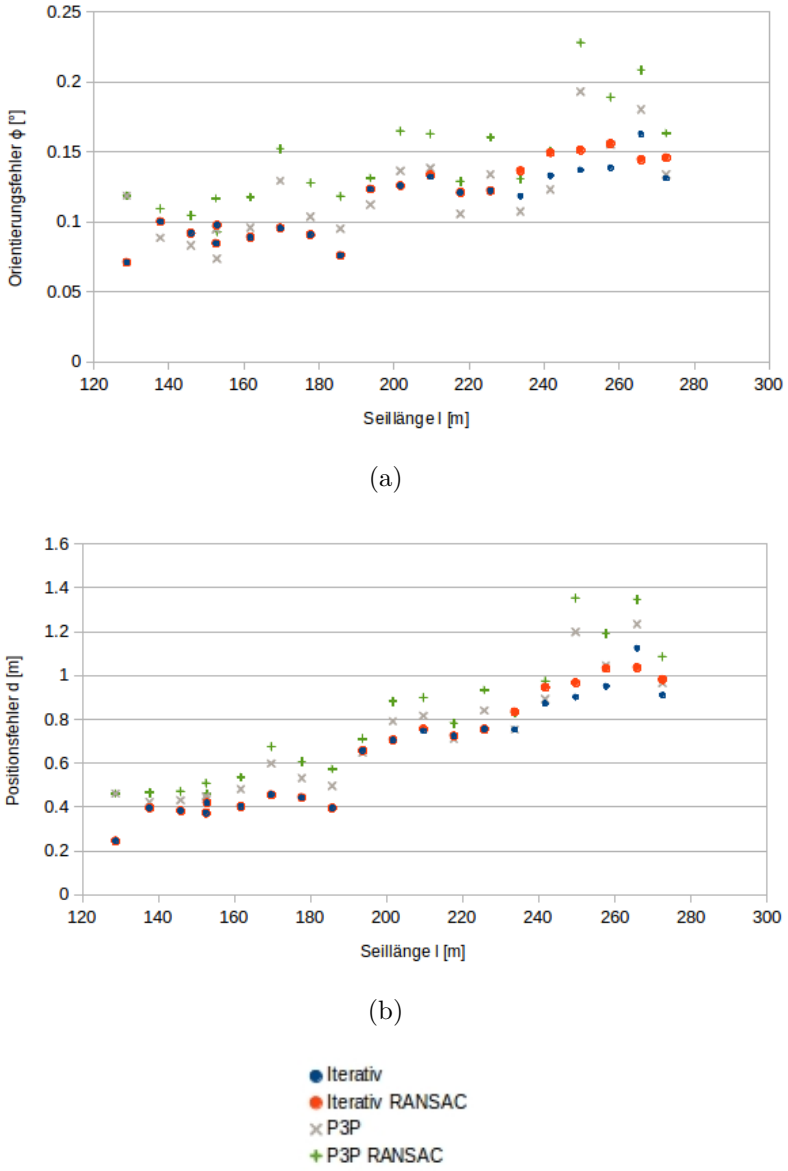


Abbildung 19: a) Orientierungs- und b) Positionsfehler der PnP-Algorithmen bei vier verwendeten Markern für die selbe Position innerhalb der Flugphase der Reel-Out-Phase der Simulation bei steigender Höhe

Aus diesem Grund kann der EPnP-Ansatz sehr gut zur genauen Bestimmung einzelner Posen genutzt werden. Für eine Nutzung während der automatisierten Posenberechnung scheint er allerdings ungeeignet. Der iterative RANSAC-Ansatz hingegen liefert eine geringfügig ungenauere, aber deutlich stabilere Orientierung: Die Orientierungsgenauigkeit ist geringfügig abhängig von der Höhe im Bereich $<0{,}1°$ und liegt im Mittel bei $0{,}061°$. Zusammen mit einer absoluten Änderung des Positionsfehlers pro Zentimeter Höhe von ca. 0,0026 cm liefert diese Methode jedoch völlig ausreichende Werte. Dies bestätigt die zuvor basierend auf den Modelltests getroffene Aussage, dass der iterative Ansatz zur Lösung des PnP-Problems in Kombination mit RANSAC die

für den konkreten Anwendungsfall besten Lösungen liefert und wird dementsprechend auch im Folgenden für Tests anhand der Simulation verwendet.

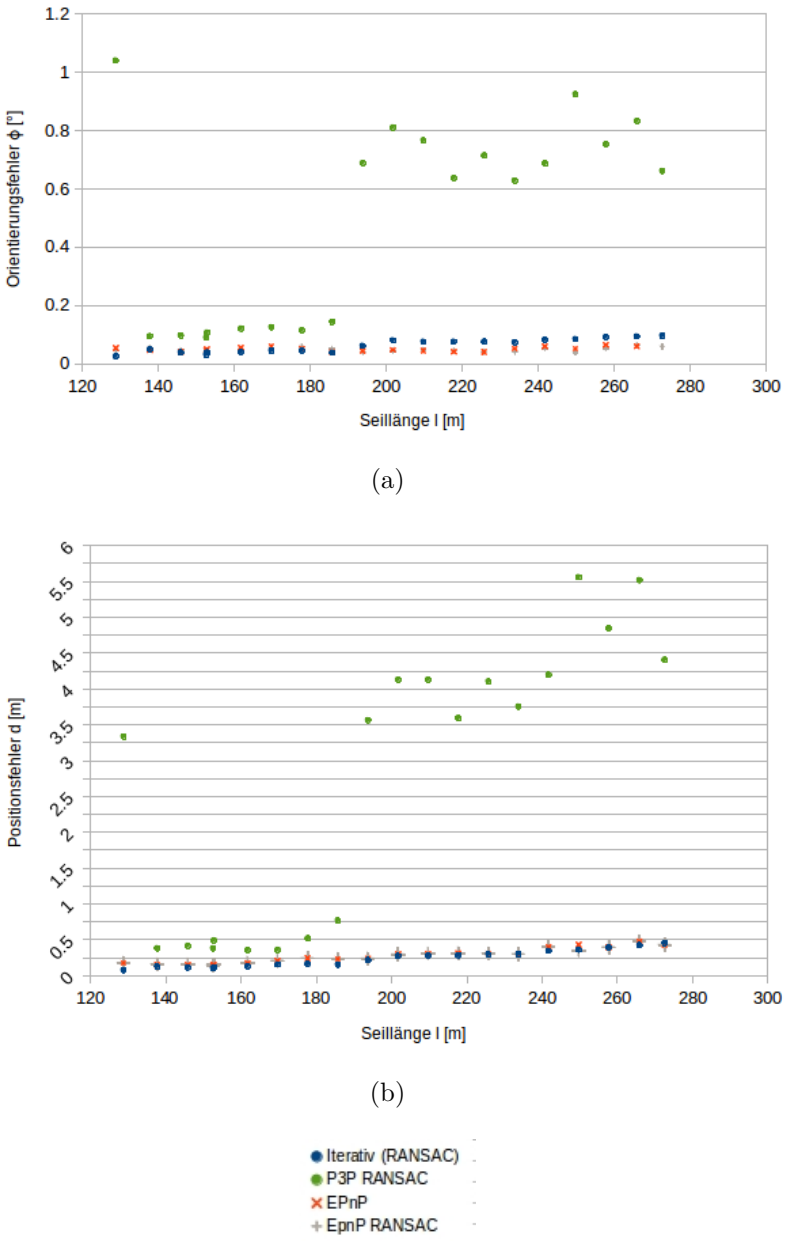


Abbildung 20: a) Orientierungs- und b) Positionsfehler der PnP-Algorithmen bei sechs verwendeten Markern für die selbe Position innerhalb der Flugphase der Reel-Out-Phase der Simulation bei steigender Höhe

Die Trajektorie der Reel-Out-Phase besteht aus zyklischen flachen Achten, die sich mit steigender Höhe wiederholen. Dabei ist der Einfluss der Position des Kites innerhalb eines Flugmusters auf die Genauigkeit der Posenbestimmung ein wichtiger Aspekt, um verlässliche Aussagen zur Pose des Kites treffen zu können. Aus diesem Grund werden drei Beispielpositionen, die in Abb. 21 dargestellt sind, innerhalb des Flugmusters gewählt und deren Positionsabweichungen, sowie durchschnittlicher und maximaler Reprojektionsfehler bei variierender Höhe von 150 cm bis 191 cm innerhalb der Reel-Out-Phase bei sechs ausgewählten Markern untersucht.

Abb. 22 zeigt die zugehörigen Graphen für Positionsabweichungen sowie durchschnittlichen und maximalen Reprojektionsfehler. Dabei ist zu erkennen, dass am Eingang der Kurve b) die Positionsabweichung größer ist. Innerhalb der Kurve wird der Winkel zwischen Kamera und der Ebene des Bodens kleiner. Das hat zur Folge, dass Abweichungen in der Bildposition der Marker eine stärkere Auswirkung auf die berechnete Pose haben, als es bei vergleichbarer Höhe aber größerem Winkel der Fall wäre. Dies ist auch bei einem Vergleich zwischen b) und c) zu erkennen. Wenngleich die Kamera bei c) zwar stark verdreht ist, deckt sie die sichtbaren Marker in einem größeren Bereich im Bild ab, als es bei b) der Fall ist. Somit wirken sich ähnlich große Reprojektionsfehler bei c) weniger stark aus. Dieser Effekt wurde in Kapitel 4.3 bereits erläutert.

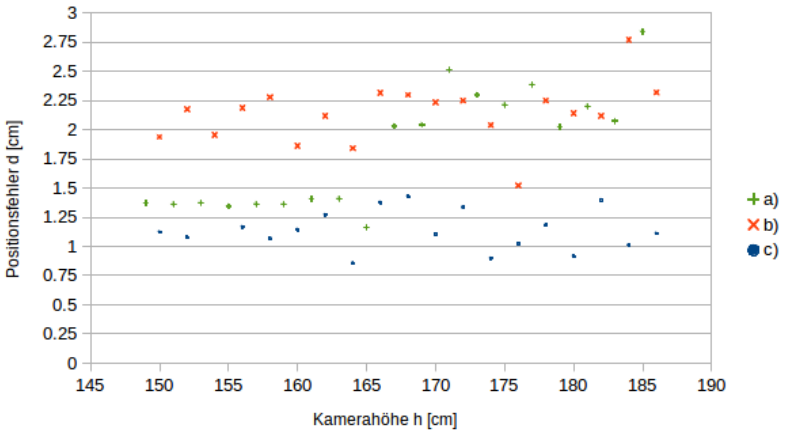
Daneben verdeutlichen die Messwerte von a), wie die Auswahl der Marker die Qualität der Messwerte beeinflussen kann: Bei einer Höhe von ca. 165 cm verschwindet ein zuvor ausgewählter Marker, da er außerhalb des Sichtfelds der Kamera liegt. Die Posenberechnung, basierend auf den verbleibenden Markern, verursacht einen Sprung in der Positionsabweichung von 1,25 cm auf über 2,5 cm. Um den starken Einfluss einzelner Marker auf die Posenbestimmung zu verhindern, bietet es sich an, möglichst viele Marker auszuwählen und den RANSAC-Algorithmus zu verwenden.

(a)

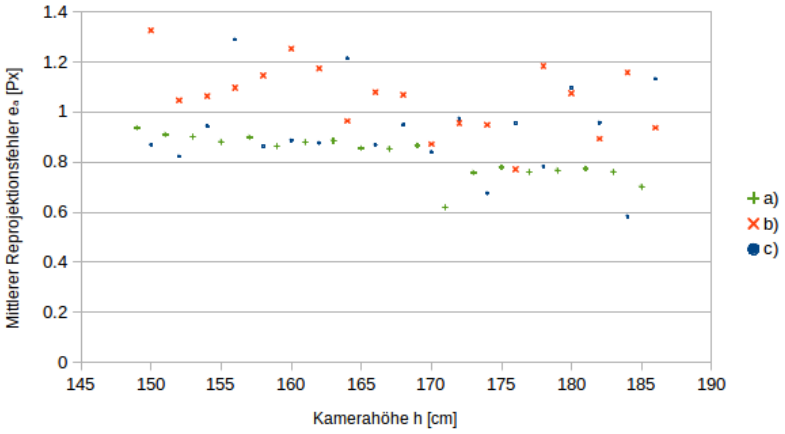
(b)

(c)

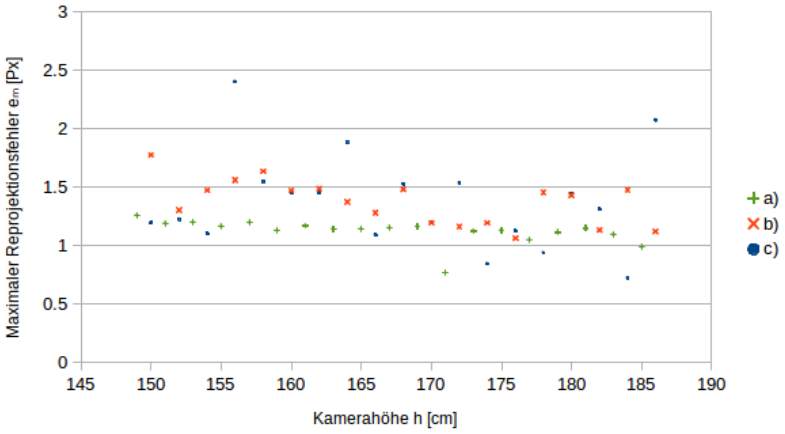
Abbildung 21: Ausgangsposition des Kites in 150 cm Höhe in a) einer zentralen Position, b) dem Eingang einer Kurve, c) dem Ausgang einer Kurve, jeweils während der Reel-Out-Phase. Das Schachbrettmuster dient zur Bestimmung der Referenzposition der Kamera und ist irrelevant für die Posenberechnung.



(a)



(b)



(c)

Abbildung 22: (a) Positionsfehler, (b) mittlerer Reprojektionsfehler und (c) maximaler Reprojektionsfehler der Beispielpositionen aus Abb. 21

Auch anhand der Simulation ist zu erkennen, dass die Abweichung der Pose abhängig von der Position des Kites innerhalb des Flugmusters ist. Dazu wird die Flugphase einer einzelnen flachen Acht simuliert und ausgewertet. Abb. 23 zeigt die entsprechende Trajektorie.

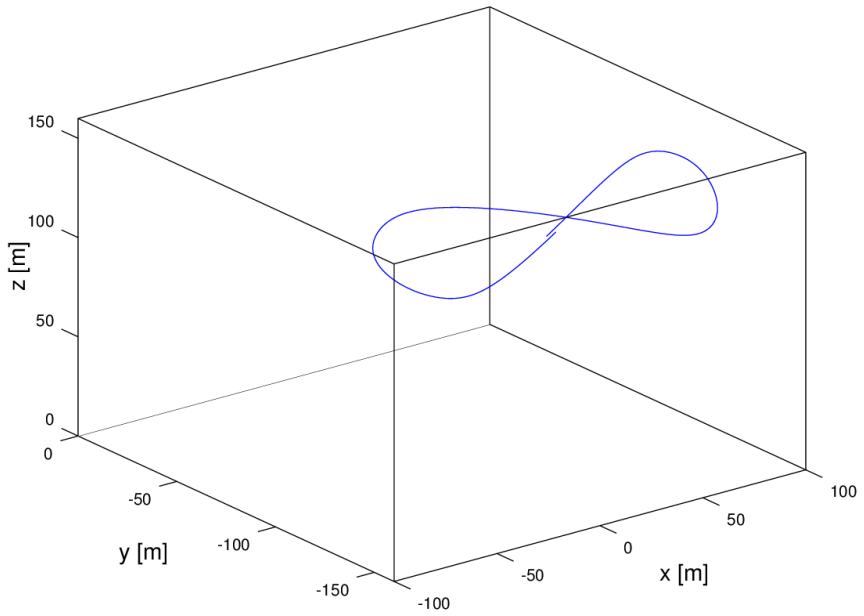
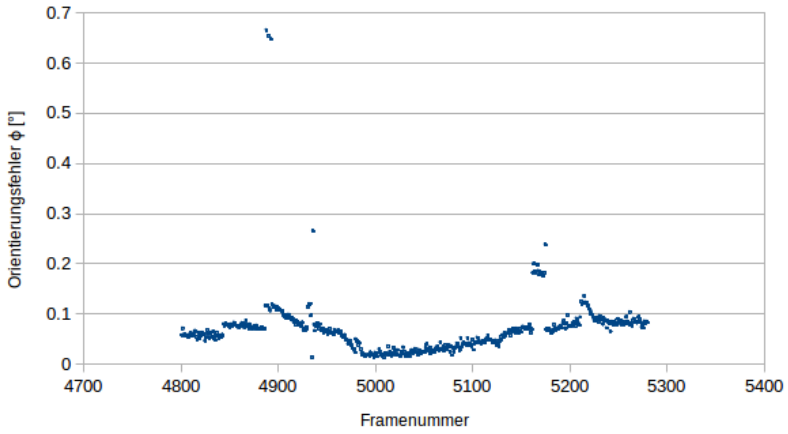
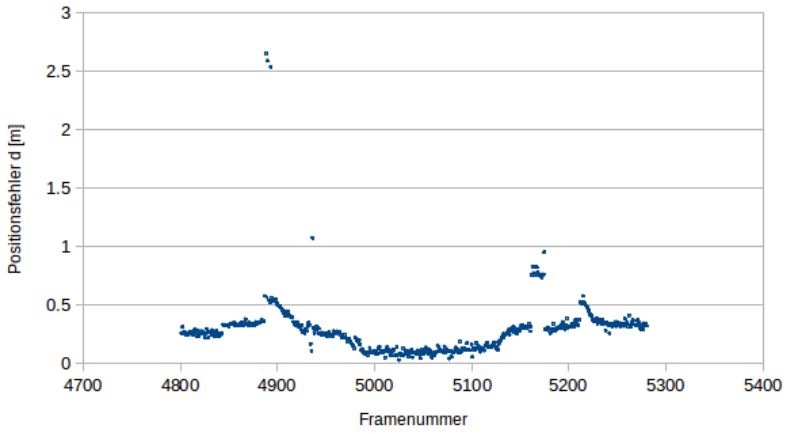


Abbildung 23: Trajektorie des Kites einer flachen Acht basierend auf den Positionsdaten der Simulation der Frames 4801 bis 5280 bei einer Seillänge von 200 bis 208 m

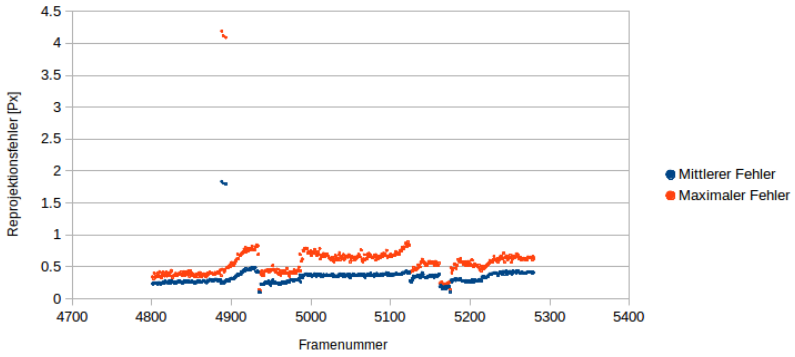
Abb. 24 zeigt den dazugehörigen Orientierungs-, Positions- sowie Reprojektionsfehler. Die Drehung des Kites findet ca. bei den Frames 4860 bis 4960 sowie 5100 bis 5240 statt. In beiden Fällen steigt sowohl der Orientierungs- als auch der Positionsfehler an. Dabei steigt der Orientierungsfehler innerhalb der Kurven ohne Berücksichtigung von Ausreißern auf bis zu 0,2°, während er in zentraler Lage des Flugmusters auf bis zu 0,017° fällt. Ähnlich verhält es sich mit dem Positionsfehler, der je nach Position innerhalb der flachen Acht zwischen 0,9 m in den Kurven und 0,07 m in zentraler Position schwankt.



(a)



(b)



(c)

Abbildung 24: (a) Orientierungs-, (b) Positions- und (c) mittlerer und maximaler Reprojektionsfehler für die Trajektorie aus Abb. 23

Außerdem ist zu erkennen, dass es vereinzelt zu Ausreißern kommt, bei denen basierend auf einem starken maximalen Reprojektionsfehler zu einer größeren Abweichung in Position und Orientierung kommt. Dies stellt jedoch kein Problem dar, da die Ausreißer problemlos nachträglich anhand des maximalen Reprojektionsfehlers gefiltert werden können. Konkret sorgen in Abb. 24 einige Ausreißer mit einem Reprojektionsfehler über 4 Px für Orientierungsfehler von bis zu 0,7° bzw. knapp unter 3 m in der Position. Alternativ kann bereits vor der Messung der maximal zulässige Reprojektionsfehler in der Konfigurationsdatei zur Posenbestimmung hinterlegt werden (siehe Kapitel 5.5), um bereits zur Laufzeit Ausreißer zu filtern. Dabei bedeutet ein geringerer Maximalwert eine häufigere Nutzerinteraktion, da der Maximalwert häufiger überschritten wird. Das Ergebnis ist jedoch eine genauere Posenbestimmung.

Da die Pose insbesondere während der Reel-In-Phase zur zuverlässigen Kontrolle des Kites benötigt wird, wird der Übergang von Reel-Out- zur Transfer- und anschließend zur Reel-In-Phase separat betrachtet (Abb. 25). Diese Phasen bestehen aus 1350 Frames, bei denen es möglich ist, einmalig im ersten Frame Marker auszuwählen und die Berechnung anschließend über alle Frames automatisiert ablaufen zu lassen. Dabei wird das Tracken der Marker mittels Lucas-Kanade-Algorithmus (siehe Kapitel 4.2) sowie das automatisierte Wiedererkennen und Schätzen der Markerposition bei kurzzeitig verdeckten Markern (siehe Kapitel 5.3) angewendet.

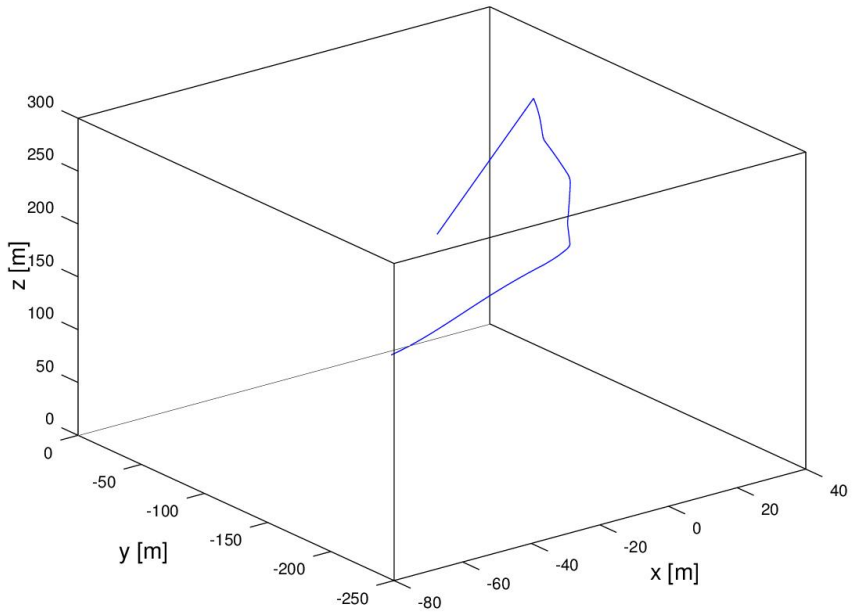
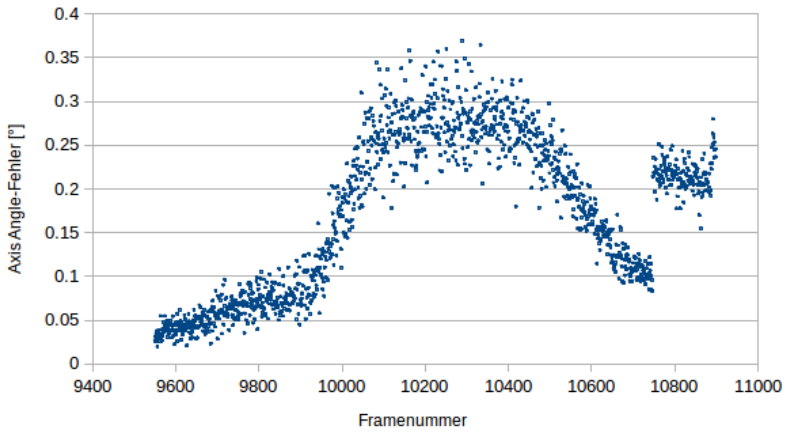
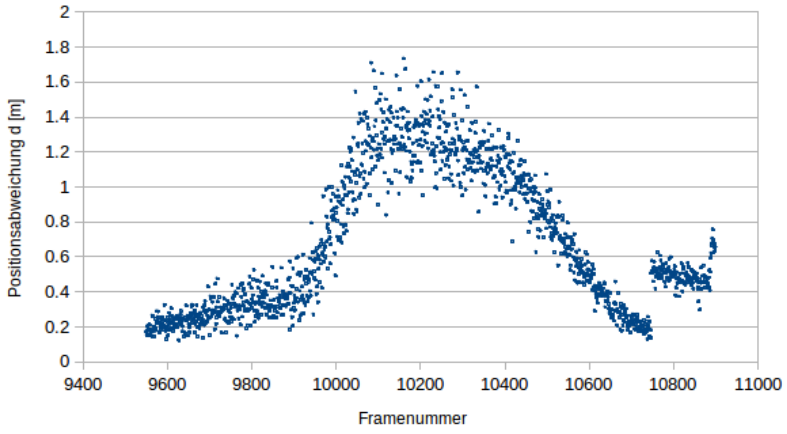


Abbildung 25: Trajektorie des Kites basierend auf den Positionsdaten der Simulation der Frames 9950 bis 10900 mit dem Übergang von Reel-Out-Phase zur Transfer- und anschließend zur Reel-In-Phase

Abb. 26 zeigt den entsprechenden Verlauf des Orientierungs- und Positionsfehlers.



(a)



(b)

Abbildung 26: (a) Orientierungs- und (b) Positionsfehler während der Reel-In-Phase. Die Berechnung lief über alle dargestellten Frames nach einmaliger Markerauswahl zu Beginn automatisiert.

Dabei steigt zunächst der Orientierungsfehler überraschend deutlich von unter 0,05° auf rund 0,37°. Dieser Anstieg ist deshalb überraschend, weil der Übergang in die Transferphase eine gute Übersicht über die Marker bietet und keine flachen Winkel bildet oder andere Störfaktoren enthält. Dies könnte durch Abweichungen in den automatisiert bestimmten Markerpositionen, die zur Berechnung der Pose genutzt werden verursacht sein. Allerdings fällt dann in der Reel-In-Phase der Orientierungsfehler stark auf rund 0,1° ab. Ein sehr ähnlicher Verlauf ergibt sich auch für die Positionsabweichung. Dort beträgt der maximale Fehler ca. 1,75 m, bevor er während der Reel-In-Phase auf bis 0,2 m abfällt. Die ermittelten Werte stellen eine stabile und definitiv ausreichend genaue Bestimmung der Pose dar. Dabei sei allerdings erneut darauf hingewiesen, dass die Simulation keine Linsenverzeichnung berücksichtigt. Es stellt sich also die Frage, ob die Kamerakalibrierung ausreichend genau möglich ist, um ähnlich gute Werte unter Verwendung der GoPro zu erhalten. Allerdings sind die anhand der Simulation ermit-

telten Daten so genau, dass auch, wie anhand des Modells gezeigt, bei einer nicht so hochwertigen Kalibrierung ausreichend genaue Werte zu erwarten sind.

Außerdem ist erneut ab ca. Frame 10800 zu erkennen, wie die Posenberechnung durch die Markerauswahl beeinflusst wird: Aufgrund der ständig sinkenden Höhe während der Reel-In-Phase sind nicht mehr alle zuvor verwendeten Marker im Bild sichtbar. Das führt im konkreten Fall zu einem Sprung des Orientierungsfehlers von ca. 0,1° auf bis zu 0,28°. Auch im Positionsfehler spiegelt sich diese Änderung wider. Dieser springt von ca. 0,2 m auf bis zu 0,8 m. Allerdings liegen diese Sprünge weiterhin weit unter einem Fehler von über 5 m oder 1° und sind somit problemlos zu verwenden.

Zunächst waren zudem Modelltests im Freien geplant, bei denen andere Umwelteinflüsse wie Sonneneinstrahlung hätten berücksichtigt werden können. Allerdings hat sich die Vermessung der Referenzpositionen mit einer ausreichenden Genauigkeit von Hand als sehr schwierig erwiesen. Da Tests bei ungenauen Referenzwerten stark an Aussagekraft verlieren, wurde auf Tests im Freien verzichtet. Für Tests in einer realistischen Umgebung würden sich weitere Hilfsmittel anbieten, die eine genaue Vermessung der Referenzwerte für Position und Orientierung ermöglichen. Dazu würde sich ein laserbasiertes Entfernungsmessgerät oder sogar Lasertracker eignen.

Auch der eigentlich naheliegende Test der Posenberechnung über alle Frames der Simulation wird nicht durchgeführt. Bei der Kombination des *Blender*-Constraints zur Kontrolle der Seillänge mit dem Constraint zur Verfolgung der flachen Acht treten Probleme in der Bestimmung der Referenzwerte der Pose auf (siehe Kapitel 6). Dort kommt es in *Blender* in den Kurven der Reel-Out-Phase der Simulation vor, dass die in *Blender* angezeigte Transformationsmatrix nicht mit der zugehörigen dargestellten Kamerapose übereinstimmt. Dabei konnte nicht mehr rechtzeitig ermittelt werden, ob es sich hier um einen internen Fehler in *Blender* oder einen selbstverschuldeten Fehler in der Umsetzung der Simulation handelt. Aus diesem Grund ist der automatisierte Test der Simulation auf die zuvor gezeigt Transfer- und Reel-In-Phase beschränkt.

Aufgrund der verschiedenen Einflussfaktoren auf die Genauigkeit der Pose wie beispielsweise der aktuellen Markerauswahl und der Genauigkeit der bestimmten Markerpositionen, ist es schwierig, eine pauschale Genauigkeit für die Posenberechnung anzugeben. Stattdessen wird ein Schwellwert definiert, der den zulässigen Bereich für eine berechnete Pose definiert. Alle durchgeführten Tests ermöglichten unabhängig von der Höhe unter Verwendung des iterativen RANSAC-Algorithmus eine Posenbestimmung mit einem maximalen Positions- und Orientierungsfehler der jeweils deutlich unter 5 m bzw. 1° lag. Dieser Wert ist ausreichend genau, um Messwerte einer IMU zu verifizieren und wird als maximale Abweichung bei der Posenberechnung angestrebt. Wenngleich im Anschluss an diese Arbeit weitere Tests nötig sind, um endgültige Aussagen über

die Qualität der Messungen machen zu können, zeigen die generierten Ergebnisse, dass die Berechnung der Pose basierend auf 2D-Luftaufnahmen definitiv möglich ist.

7.2 Weitere Anmerkungen

Im Folgenden sollen kurz weitere Tests erläutert werden, für die keine Daten grafisch aufbereitet werden, die aber dennoch relevant für die Funktionsweise des Programms sind. Bei den Tests der Pose ist aufgefallen, dass die Marker zum Teil nicht automatisiert über eine Kantendetektion in der Region des Nutzerinputs erkannt werden konnten. Um die Ursache zu untersuchen, wurde zunächst die Bildposition eines Markers als Reprojektion aus den Posendaten ermittelt und anschließend der Marker in 15°-Schritten rotiert. Anschließend wurde die zuvor ermittelte Bildposition mit der mittels Shi und Tomasi bzw. Harris detektierten Position verglichen. Dabei zeigt sich, dass die Marker nicht erkannt werden, wenn ihre inneren Kanten parallel zu den Bildrändern verlaufen. Die Marker sollten aus diesem Grund mit variierender Ausrichtung platziert werden, damit immer einige von den Kantendetektoren erkannt werden können. Dabei hat die Orientierung keine Auswirkung auf das Tracken der Marker, sondern lediglich auf das Detektieren der Kante bei der initialen Auswahl eines Markers. Dabei kann bei Bedarf immer die konkrete Markerposition vom Nutzer mit einem Rechtsklick fociert werden (siehe Kapitel 5.2).

Zum allgemeinen Test des Markertrackings wurde zudem das entwickelte Programm mit dem originalen Video des SkySails-Prototypenflugs gestartet und markante Objekte wie beispielsweise Autodächer oder Kanten der Bodenstation ausgewählt. Diese Punkte konnten problemlos verfolgt werden, sodass davon ausgegangen werden kann, dass auch das Verfolgen der markanteren Marker kein Problem darstellt. Dabei konnten bei diesem Test keine Daten generiert werden, da die benötigte 3D-Position der ausgewählten Punkte im Bild für eine Posenberechnung nicht bekannt war. Bei Trackingversuchen hat sich anhand der Simulation gezeigt, dass Lichteffekte in der Regel keinen Effekt auf das Verfolgen der Marker haben. Dies sollte insbesondere bei realen Aufnahmen kein Problem darstellen, da der verwendete Stoff kaum Licht reflektiert. Auswirkung auf das Tracken der Marker haben hingegen ruckartige Bewegungen. Diese in der Simulation künstlich erzeugten Bewegungen waren jedoch so stark, dass sie im Regelfall nicht ohne Fremdeinwirkung am Kite auftreten sollten.

Zum Testen des Trackings anhand des Modells wurden zudem Aufnahmen gemacht, in denen die Kamera um ca. 60° um die einzelnen Achsen rotiert wurde. Dabei konnten die Marker korrekt verfolgt werden. Allerdings ist es erneut nicht ohne Weiteres möglich, von Hand die zur grafischen Darstellung benötigten Refernzwerte der Kamerafahrt, insbesondere der Orientierung, zu ermitteln. Allerdings stellt der Tests der Reel-In-Phase anhand der Simulation die

8 Fazit

Es gibt viele Ansätze zur Berechnung der Pose eines Objekts, die für verschiedene Anwendungsfälle ausgelegt sind. Im Rahmen dieser Arbeit stand die Bestimmung der Pose basierend auf 2D-Luftaufnahmen im Vordergrund.

Dabei wurde die Frage, ob es mit vertretbarem Aufwand möglich ist, die Pose eines fliegenden Kites basierend auf 2D-Luftaufnahmen ausreichend genau zu bestimmen, beantwortet: Sowohl anhand der entwickelten Simulation als auch anhand der Modellaufnahmen ist es gelungen, die Pose für die einzelnen Frames des Videos ausreichend genau zu bestimmen.

Ausschlaggebend für die Genauigkeit der berechneten Pose ist dabei vor allem die genaue Kenntnis sowohl der 2D-Bildposition als auch der realen 3D-Markerposition. Aus diesem Grund sollte bereits bei der Auslegung des Koordinatensystems der Marker am Boden besonders sorgfältig vorgegangen werden, um einen – nachträglich nicht zu korrigierenden – Fehler in der 3D-Markerposition zu verhindern.

Die Abweichung in der Berechnung ist außerdem besonders abhängig davon, welche Marker aktuell für die Berechnung ausgewählt sind. Dabei hat sich gezeigt, dass die besten Ergebnisse erzielt werden, wenn möglichst viele Marker für die Berechnung ausgewählt sind. Das PnP-Problem kann dann iterativ unter Verwendung des RANSAC-Algorithmus für eine stabile und zudem genaue Posenberechnung genutzt werden, die aufgrund der Vielzahl an verwendeten Markern zudem eine nötige Interaktion durch den Nutzer minimiert.

Aufgrund der vielen Einflussfaktoren ist es nur schwierig möglich, eine absolute Genauigkeit für die Pose anzugeben. Es bietet sich daher an, einen Schwellenwert zu definieren, den die berechneten Posenwerte für alle Tests nicht überschreiten. Dieser maximale Wert wird unabhängig von der Höhe für den Positions- auf 5 m und die Orientierung 1° festgelegt. Alle Tests liegen deutlich unterhalb dieser Werte, sodass auch schlechtere Messungen noch in diesem Rahmen liegen sollten. Dabei wurde die Erwartung bestätigt, dass vor allem die Größenordnung des Positionsfehler von der Höhe des Kites abhängig ist. Die Größe des Fehlers ist dabei direkt abhängig von der Kamerakalibrierung. Um die Posenberechnung bei Bedarf weiter zu verbessern, empfiehlt es sich deshalb, einige Zeit in die Kalibrierung der Kamera zu investieren.

Der präsentierte Ansatz ist für die Verifikation der Pose für zuvor generierte Daten geeignet. Allerdings beschränkt sich dies auf eine nachträgliche Verifikation zuvor gewonnener Daten. Inbesondere die fehlende Echtzeitfähigkeit sowie die Anfälligkeit gegenüber Wettereinflüssen und die vorausgesetzte ständige Sichtbarkeit der Marker schließen eine Nutzung der Posenbestimmung mittels Bildverarbeitung als Hauptkomponente in der Regelung des Kites aus. Dies wird zusätzlich durch den umgesetzten

semiautomatisierten Ansatz beschränkt: Während des Flugs ist ein Eingreifen durch den Nutzer zur Korrektur der Markererkennung unmöglich und somit nicht umzusetzen.

Denkbar wäre in Zukunft jedoch beispielsweise ein Ansatz, in dem die Pose mithilfe der 2D-Luftaufnahmen zyklisch mit einer geringeren Frequenz bereits während des Flugs berechnet und als Referenzwert der IMU zur Verfügung gestellt wird, um den über die Zeit akkumulierten Drift der IMU zu korrigieren. Dazu wären jedoch zunächst weitläufige Tests insbesondere des Markertrackings nötig, da die Posenberechnung dann vollständig automatisiert ablaufen und verdeckte Marker korrekt wiedererkannt werden müssten. Demgegenüber steht die Option, in Zukunft die Genauigkeit der Pose weiter zu optimieren und weiterhin für eine Verifizierung der Daten der IMU zu nutzen. Dazu würde es sich anbieten, auf Kosten der Rechenzeit die verschiedenen Algorithmen zur Posenbestimmung innerhalb von OpenCV zu kombinieren. Dabei müsste zunächst jedoch geklärt werden, ob eine genauere Pose des Kites einen Mehrwert für dessen Regelung mit sich bringen würde.

Persönlich habe ich die Bildverarbeitung immer als Diziplin gesehen, die sich auf stark spezialisierte Bereiche wie die Robotik beschränkt. Im Rahmen dieser Arbeit habe ich gelernt, dass Methoden der Bildverarbeitung heutzutage durchaus in der Lage sind, kostengünstige Lösungen von Problemen zu liefern, die im Regelfall mit herkömmlicher Sensorik gelöst werden würden.

Insgesamt lässt sich der Schluss ziehen, dass die vorgestellten Methoden der Bildverarbeitung gut geeignet sind, bestehende Messsysteme zur Bestimmung von Position und Orientierung zu ergänzen, dass sie diese aber definitiv nicht gänzlich ersetzen können.

Literaturverzeichnis

[Age15] Agentur für Erneuerbare Energien, BDEW: *Strommix in Deutschland 2014*. `http://www.unendlich-viel-energie.de/strommix-deutschland-2014`. Version: Januar 2015

[Bou00] Bouguet, J.: Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. (2000)

[Die13] Diehl, M.: Airborne Wind Energy: Basic Concepts and Physical Foundations. In: Ahrens, U. (Hrsg.) ; Diehl, M. (Hrsg.) ; Schmehl, R. (Hrsg.): *Airborne Wind Energy*. Springer, 2013, Kapitel 1, S. 3–22

[ES13] Erhard, M ; Strauch, H.: Sensors and Navigation Algorithms for Flight Control of Tethered Kites. In: *2013 European Control Conference (ECC)*. Zürich, Schweiz : IEEE, 2013, S. 998–1003

[ES15] Erhard, M. ; Strauch, H.: Flight Control of Tethered Kites in Autonomous Pumping Cycles for Airborne Wind Energy. In: *Control Engineering Practice* 40 (2015), S. 13–26

[FB81] Fischler, M. A. ; Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395

[FMSS14] Faessler, M. ; Mueggler, E. ; Schwabe, K. ; Scaramuzza, D.: A Monocular Pose Estimation System Based on Infrared LEDs. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference*. Hong Kong : IEEE, 2014, S. 907–913

[GHTC03] Gao, X. ; Hou, X. ; Tang, J. ; Cheng, H.: Complete Solution Classification for the Perspective-Three-Point Problem. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions* 25 (2003), Nr. 8, S. 930–943

[HS88] Harris, C. ; Stephens, M.: A Combined Corner and Edge Detector. In: *Proceedings of the Alvey Vision Conference*, Alvety Vision Club, 1988, S. 147–152

[Huy09] Huynh, D.: Metrics for 3D Rotations: Comparison and Analysis. In: *Journal of Mathematical Imaging and Vision* 35 (2009), Nr. 2, S. 155–164

[LMNF09] Lepetit, V. ; Moreno-Noguer, F. ; Fua, P.: EPnP: An Accurate O(N) Solution to the PnP Problem. In: *Int. J. Comput. Vision* 81 (2009), Februar, Nr. 2, S. 155–166

[MCMOM10] Mondragon, I. F. ; Campoy, P. ; Martinez, C. ; Olivares-Mendez, M. A.: 3D Pose Estimation Based on Planar Object Tracking for UAVs Control. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference*. Anchorage, AK, USA : IEEE, 2010, S. 35–41

[MKNK14] Munkelt, C. ; Kühmstedt, P. ; Notni, G. ; Kleiner, B.: Navigationsbasierte automatische Multi-View-3D-Messung. In: *Fraunhofer IOF Jahresbericht 2014* (2014), S. 84–85

[Ope15] OpenCV Dev Team: *Camera Calibration and 3D Reconstruction*. `http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`. Version: Februar 2015

[Sim13] Simek, Kyle: *Dissecting the Camera Matrix*. `http://ksimek.github.io/2012/08/14/decompose/`. Version: August 2013

[SMB00] Schmid, C. ; Mohr, R. ; Bauckhage, C.: Evaluation of Interest Point Detectors. In: *International Journal of Computer Vision* 37 (2000), Nr. 2, S. 151–172

[ST94] Shi, J. ; Tomasi, C.: Good Features to Track. In: *Computer Vision and Pattern Recognition, 1994*, IEEE, 1994, S. 593–600

[Woo07] Woodman, O. J.: An Introduction to Inertial Navigation / University of Cambridge Computer Laboratory. Cambridge, UK, 2007 (696). – Forschungsbericht

[Zha98] Zhang, Zhengyou: A Flexible New Technique for Camera Calibration / Microsoft Research. Redmond, USA, 1998 (71). – Forschungsbericht

Anhang

Der Anhang befindet sich in digitaler Form auf der beigelegten CD inklusive einer digitalen Auflistung der vorhandenen Daten und Dokumente.