Praktikum zur Bachelorarbeit

Nina Suette, 11912463

3D zu 2D Drohnenflug Visualisierung

Im Folgenden wird das Projekt für die Lehrveranstaltung "Praktikum zur Bachelorarbeit" im Wintersemester 2021/21 dokumentiert. Das Projekt wird von der Studentin Nina Suette an der AAU Klagenfurt bearbeitet. Die Gesamtaufgabe besteht darin, eine von der einer Drohne geflogenen, mit Optitrack aufgenommenen Trajektorie direkt im Bild zu visualisieren. Dies soll durch das Kalibrieren der Kamera und das Transformieren der 3D Punkte der realen Welt zu 2D Punkten in der Bildebene realisiert werden.

1 Bestandteile

Um das Projekt zu realisieren, müssen grob gegliedert folgende Ressourcen vorhanden sein:

- Optitrack für Motion Capture
- Kamera mit Marker
- Kalibrierungsmuster (Kalibr Docker)
- Ubuntu2004 (Oracle VM VirtualBox)
- Robot Operating System (ROS)
- Matlab (optional)

Die Autorin des Dokuments hat als Programmiersprache C++ verwendet, zusätzlich wurden für die Umsetzung des Projektes Ubuntu 20.04 Virtualbox als Betriebssystem verwendet. Das System Robotic Operating System (ROS) wurde eingesetzt, um die Kommunikation zwischen Prozessen zu verwirklichen. Zusätzlich wurde Open Computer Vision, OpenCV als Open-Source-Softwarebibliothek genutzt. Das Programm MATLAB wurde für das Generieren der Trajektorien eingesetzt.

2 Ordnerstruktur

Für das Kalibrieren der Kamera und Für das projizieren der Trajektorie enthält die vorliegende Ordnerstruktur jeweils eigene Verzeichnisse. Für die Kalibrierung der Kamera wird das Ordnerverzeichnis docker_kalibr verwendet, für die Projektion der Matrix wird das Verzeichnis catkin_ws verwendet, welches gleichzeitig der Workspace des Projektes und somit Hauptverzeichnis ist. Im Projekt ist die Ordnerstruktur wir folgt definiert:

- docker_kalibr
    - target_a3.pdf: Kalibierungsmuster
    - camera_calibration_ids_1
        - intrinsic: Kalibrierungsdaten intrinsische Parameter
        - extrinsic: Kalibrierungsdaten extrinsische Parameter
- catkin_ws
    - build
    - devel
    - install
    - src
        - aaucns_trajectory_projector
            - trajectory_projector
                - bag: BAG-File mit Videomaterial
                - cfg
                - include: Header für das Hauptprogramm
                - launch: Launch-File Configurationen
                - msg: Massage Configurationen
                - src
                    - main.cpp
                    - trajectory_projector.cpp: Hauptprogramm
                - srv
                - CMakeLists.txt
                - package.xml
                - TrajectoryProjectorParams.yaml: Kameraparameter

3 Kalibrierung der Kamera

Ziel der Kamera Kalibrierung ist es, die intrinsischen Kameraparameter, die extrinsischen Kameraparameter und die Verzerrungskoeffizienten für die verwendete Kamera ausfindig zu machen. Im durchgeführten Experiment wurde für die Kalibrierung eine Kamera des Modells "pinhole-radtan" mit der Auflösung von 2056x1542 verwendet, "pinhole" bedeutet, dass es sich um eine Lochkamera handelt, "radtan", dass mit radialen und tangentialen Verzerrungs-koeffizienten gearbeitet wird.
Für die Umsetzung der Kalibration wird ein Tool verwendet, welches sich Kalibr Docker Image nennt. Dieses Tool wird eingesetzt, um zu vermeiden, dass individuell alle Library dependencies installiert werden müssen, oder Software Versionen inkompatibel werden (z.B. OpenCV in Ubuntu 20.04). Dieser Docker Container bietet somit die Möglichkeit, ein virtuelles System lediglich mit einer Kalibr Installation zu verwenden. Einfachheit halber wird um Daten zwischen dem Host System und dem Docker Container auszutau-schen, Volume-Mapping verwenden. Damit können alle Daten aus dem Ordner, in dem der Docker ausgeführt wird, im Ordner innerhalb des Docker Containers gefunden werden. Diese Daten werden Bi-Direktional im Container und auf dem Host System geändert.
Zusätzlich werden noch ein Schachbrettmuster, sowie ein BAG-File mit Videomaterial, auf welches später auch projiziert werden kann, verwendet. Folgend können mit dem Kalibr Tool zuerst die intrinsischen und dann die extrinsischen Parameter generiert und automatisch in einem File abgespeichert werden, aus welchem die entstandenen Parameter im folgenden Schritt zu entnehmen sind. Die entnommenen Parameter werden händisch in ein YAML-File übertragen, welches später im Code eingelesen werden kann.

Erhalt intrinsische Kameraparameter mittels Kalibr Docker:

```
cd docker_kalibr/camera_calibration_ids_1/intrinsic

docker run -it --rm --net=host --env="DISPLAY"
--volume="$HOME/.Xauthority:/root/.Xauthority:rw" -v "$(pwd)":/kalibr/data
christianbrommer/kalibr:latest

kalibr_calibrate_cameras --topics /mission_cam/image_raw --models pinhole-radtan
--target ./target_a3.yaml  --bag ./merged.bag # --show-extraction
```

Zwei files werden generiert: camchain-.merged.yaml und results-cam-.merged.txt, aus diesen können die intrinsischen Kameraparameter und die Verzerrungskoeffizienten entnommen und in das File TrajectoryProjectorParams.yaml einfügt werden. Im Anschluss muss das camchain-.merged.yaml in den extrinsic Ordner kopieren, um es dort verwenden zu können.

Erhalt extrinsischen Kameraparameter mittels Kalibr Docker:

```
cd docker_kalibr/camera_calibration_ids_1/extrinsics

docker run -it --rm --net=host --env="DISPLAY"
--volume="$HOME/.Xauthority:/root/.Xauthority:rw" -v "$(pwd)":/kalibr/data
christianbrommer/kalibr:latest

kalibr_calibrate_imu_camera --target target_a3.yaml --cam camchain-.merged.yaml
--imu imu_model_px4.yaml --bag ./merged.bag # --show-extraction
```

Drei files werden generiert: camchain-imucam-.merged.yaml, imu-.merged.yaml und results-imucam-.merged.txt. Im camchain-imucam-.merged.yaml File ist die T_cam_imu Matrix zu finden, welche die homogene Transitionsmatrix ist. Sie setzt sich aus den Rotations- und den Translationsmatrizen zusammen. Diese Parameter werden ebenfalls im File TrajectoryProjectorParams.yaml eingefügt.

4 Programmcode

Das Herzstück des Programmcodes stellen die beiden C++-Datein trajectory_projector.cpp im src Ordner und trajectory_projector.h im include Ordner dar. In der trajectory_projector.h Datei befinden sich Deklarationen und includes. In der trajectory_projector.cpp Datei befindet sich der Programmcode, in welchen die Kameraparameter einegelesen werden. Im Hauptcode wird auch die Trajektorie, welche eine Liste aus Koordinatenpunkten ist, eingelesen und in eine Matrix transformiert.
Um die aktuellen Rotations- und Translationsparameter zu erhalten wird ein Subscribe zu einem Topic gemacht, worauf im nächsten Kapitel näher eingegangen wird. Außerdem bietet das Hauptprogramm eine Transformation der dreidimensionalen Koordinaten der Trajektorie in zweidimensionale Bildkoordinaten, welche direkt auf die Bildebene abgebildet werden können.
Umsetzung der Transformation erfolgt mit Verwendung der Open ComputerVision (OpenCV) Open-Source-Softwarebibliothek. Andere wichtige Bestandteile sind die Lauch-Datei trajectory_projector.launch im launch Ordner, welche in der Sektion "Das Launch File" genauer beschrieben wird, die package.xml Datei und in der CMakeList.txt Datei, der bag Ordner, in welchem sich die Bagfiles mit dem Videomaterial zum Kalibrieren und Projizieren befinden, sowie das DoubleArrayStamped.msg File im msg Ordner, welches das Messageformat bestimmt.

5 An ein Topic subscriben

Zunächst muss herausgefunden werden, an welches Topic man subscriben will, mit diesem Befehl können Topics aufgelistet und relevante Topics herausgesucht werden:

```
rosbag info <bag>.bag
```

Um herauszufinden, welche Topics relevant sein könne kann folgender Befehl verwendet werden:

```
rostopic list
```

Wurde ein releventes Topic ausgewählt so können wie folgt, Informationen über diese Topic herausgefunden werden:

```
rosmsg show geometry_msgs/TransformStamped
```

In diesem Beispiel werden Informationen für das für deses Projekt relevante Topic geometry_msgs/TransformStamped augegeben.Aus diesen Inforamtionen können nun Parameter und deren Datentyp abgelesen, werden welche in einer .msg Datei erfasst werden können, um im Code die aktuellen Rotations- und Tanslationsparameter zu erhalten.

im Header muss das Topic, welches verwendet wird includieren:

```
#include <geometry\_msgs/TransformStamped.h>
```

Ebenso müssen die im Hauptprogramm verwendete Methode und Parameter im Header definiert werden. Im Hauptprogramm kann nun mittels Nodehandler an das Topic subscribed werden.Nun können die Parameter mittles der definieten message im Code verwendet werden.

subscribe:

```
nh.subscribe("/twins_three/vrpn_client/raw_transform", 1,
&TrajectoryProjector::actualRotTrans, this);
```

auslesen der Parameter:

```
msg->transform.translation.x
```

Zusätzlich ist noch zu beachten, dass folgende Zusätze in der package.xml Datei und in der CMakeList.txt Datei vorgenommen werden.

package.xml Datei:

```
<build_depend>geometry_msgs</build_depend>
<run_depend>geometry_msgs</run_depend>
```

CMakeList.txt Datei:
- in find_package() das Topic hinzufügen
- in generate_messages() dtas Topic hinzufügen
- in catkin_package() das opic hinzufügen
- in add_message_files() das Message file hinzufügen

6 Das Launch-File

Im Launchfile werden der Name des auzuführenden Programmes, der Packagename, das relevante Topic und die .yaml Datei, welche die Kameraparameter enthält, definiert, Änderungen sind hier vorzunehmen, wenn beispielsweise das Bag-File geändert werden soll.

7 Das Programm starten

Sind alle Bestandteile des Projektes vollständig, so kann das Programm gestartet und die im Code fix eingelesene Trajektorie auf das Bag-File (oder auch live auf das Kamerabild) projiziert werden.

Wurde das package noch nicht erstellt, oder wurden zwischen, dem letzen "make" Änderungen im Programmcode vorgenommen, so wird vor dem Auführen des Programms ein erneutes "make" empfohlen. Dazu muss in den Ordner, in dem sich der workspace befindet, navigiert werden:

```
cd catkin_ws
catkin_make
```

1.Terminal: Damit alle Bestandteile im Package miteinander kommunizieren können, muss zuerst der Master gestartet werden:

```
roscore
```

2.Terminal: Start des roslaunch servers:

```
roslaunch trajectory_projector tarjectory_projector.launch
```

3.Terminal: image publish:

```
rosrun image_view image_view image:=trajectory_projector/image_publish
```

4.Terminal: Bag-File abspielen:

```
cd catkin_ws
cd src
cd auucns_trajectory_projector/
cd tarjectory_projector/
cd bag
rosbag play merged.bag
```

8 Ausgabe

Nach Ausführen der in Section 7 beschriebenen Befehle, öffnet sich ein Fenster, in welchem die Aufnahme des Videomaterials des Bag-Files zu sehen ist. Die in den Code eingelesene Trajektorie, bestehend aus dreidimensionalen Weltkoordinaten, kann nun dur die Transformation zu zweidimensionalen Bildkoordinaten, direkt mit dem Kamerabild überlagert weden. Die Aufnahmen machen den Anschein, die Tajektorie wirklich im Raum sehen zu können. Noch deutlicher wird dieses Ergebnis, wenn die Trajrktorie live auf das Kamerabild projiziert wird, hier kann die Tarjektorie von allen Seiten betrachtet werden.