Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robotic Systems Lab
Prof. Dr. Marco Hutter

Semester Thesis

Handheld Augmented Reality for Robotic Excavators

Autumn Term 2023

Supervised by:
Dominic Jud
Ryan Luke Johns

Author:
Lasse Fierz

Contents

Preface

The aspiration for this project was to contribute to the process of making use of new technology in order to assist humankind. The field of robotics has tremendous potential in this regard and I feel fortunate to be partaking in this endeavour.

Abstract

This project aimed at paving the way for the creation of a high level remote control for autonomous excavators using a handheld augmented reality device. The goal was to provide the necessary building blocks for a handheld remote control to fullfil the task of feeding an input to the excavator. The main requirements needed are a way of sharing information between the two devices and solving the colocalization problem between the reference frames. After failed previous attempts of establishing the data transfer using a ROS connection between the two devices the focus in this work lied on making use of the already implemented excavator setup.

Chapter 1

Introduction

When operating an excavator in the conventional fashion two requirements are imperative:

- The view onto the construction site
- The handles necessary to give inputs to the machine

In the case of an autonomous excavator the low level actions to perform are determined by the machine itself, given a high level action input. So in this case the second requirement becomes the possibility to give such a high level input to the system.
With the desired handheld remote control setup the necessity of sharing information between the device and the machine still persists. The visual requirements change however. From the handheld camera we now receive the required view of the surroundings on the construction site but in order to successfully share a geometric location with the excavator the colocalization problem has to be solved for the two players. Having an AR remote control integrated in a handheld device also provides the possibility of introducing further useful features such as displaying a preview of an action of choice.
In this project I attempted to overcome the key challenges constituting the requirements mentioned above.
The plan was to solve the data transfer requirement utilizing the already implemented Unreal Engine setup of the autonomous excavator[1] from the Robotic Systems Laboratory. To account for the colocalization problem an approach using Microsoft's Azure Spatial Anchors was used.

[1]HEAP - The autonomous walking excavator[1]

Chapter 2

Related Work

The work related to this thesis can be divided into two categories:
First there is the traditional 2D computer vision aspect to it for which I can't reference a specific work as the field is just too broad. However some crucial mentions are the different feature methods that I considered in this work (ORB[1], BRISK [2] and KLT[3]) as well as the outlier rejection procedure RANSAC[4].
The second part concerns the 3D side of the paper being state of the art LiDAR usage for motion estimation:
LiDAR as a tool is of course no new idea and many ingenious people have already ventured into this field and refined methods to work with the 3D data that LiDAR provides us with. The traditional procedure for achieving motion estimation using LiDAR is point cloud registration and there have been a lot of papers published about this idea. One that I would like to point out is the paper of Pomerleau et al. [5] which summarizes the ICP algorithm (Iterative Closest Point) as well as certain usage cases.
An alternative procedure to estimate the motion and construct a map of the surroundings at run time is LOAM [6] which is built around the idea of splitting the two algorithms up into the odometry and the mapping part. For the odometry part the detected features are divided into planar patches and sharp edge lines which are then used to establish correspondences and thus achieve motion estimation. The mapping process makes use of the iterative scans as well as the transformations each step in order to build the permanent map in the world frame. This map in turn can be consulted in order to achieve much more accurate motion estimation.

This bachelor thesis however was built on the idea of projecting dense point clouds of newer LiDAR scans onto planes and performing state of the art 2D CV methods on the projections as opposed to applying computationally expensive alignment methods. I thus used the best of both worlds by achieving run time performance without neglecting a significant amount of information through downsampling the point clouds.

[1]Oriented FAST and Rotated BRIEF [**?**]
[2]BRISK: Binary Robust Invariant Scalable Keypoints [**?**]
[3]Lucas Kanade Tracking [**?**]
[4]RANdom SAmple Consensus[**?**]
[5]A Review of Point Cloud Registration Algorithms for Mobile Robotics [**?**]
[6]LiDAR Odometry and Mapping [**?**]

With great probability the first paper published about this new way of working with LiDAR data is the paper of Shan et al. [7] In their work they used this approach in order to extract ORB features each scan and to build up a BoW database which they queried in order to find matches with later extracted features to determine loop closures. In comparison to their work I used this underlying method in order to achieve motion estimation considering just two subsequent scans as well as different descriptors and complementary types of data.

[7]Robust place recognition using an imaging lidar [**?**]

Chapter 3

Method

As mentioned above the approach in this project can be divided into two stages:

- Data transfer between the excavator and the handheld device
  - Making use of the already existing UE interface on the excavator to store system information in game components
  - Creating an Augmented Reality Unreal Engine (UE) game on the handheld device
  - Using an Unreal Engine local multiplayer connection in order to transfer data between the two UE instances
- Colocalization between the two frames of reference
  - Extracting a spatial anchor from the excavator's camera view using a ROS wrapper
  - Uploading this visual anchor in the form of an Azure Spatial Anchor to the Azure cloud
  - Retrieving the stored spatial anchor in the handheld UE instance
  - Using the spatial anchor to relocate the UE world origin

Excavator Setup

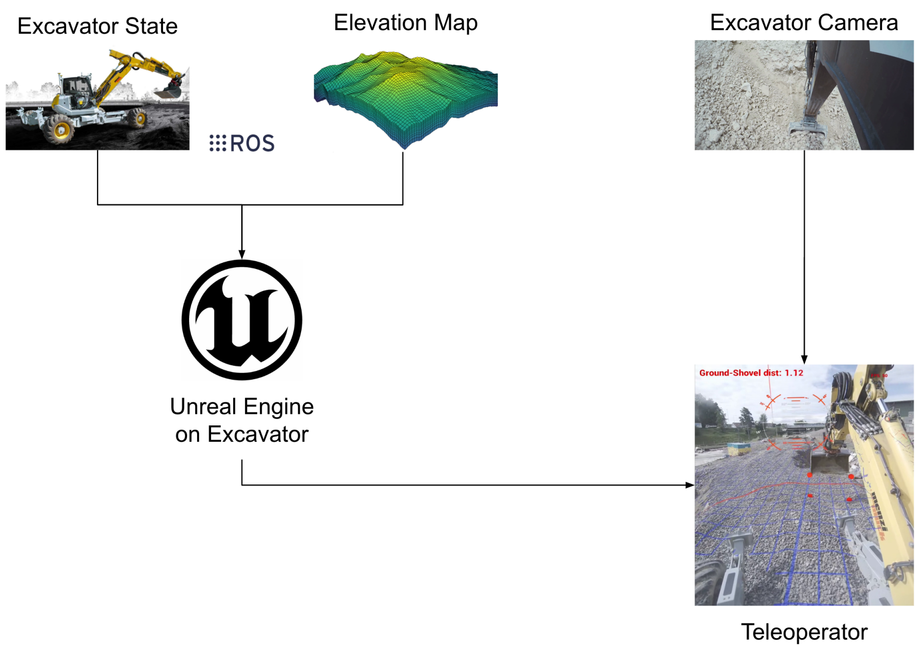

Figure 3.1: High level excavator setup prior to this project

The excavator has an Unreal Engine instance running on it. In the UE instance there is a model of the excavator which is updated continuously using a built in ROS subscriber in UE.
Another crucial input to the excavator is an elevation map which is generated from the visual input of the environment.
Through the current state of the excavator at any time it is possible to calculate the parts in the camera input image which are occluded by the excavator. This feature allows for the creation of a mask to cut the excavator out of the elevation map projection.
In the end the Unreal Engine instance outputs this tailored elevation map projection which can be applied on top of the input image which is fed to the teleoperator.
Having a running simulation of the excavator in the UE instance also allows for more useful features such as the excat calculation of the shovel position and the projection thereof onto the ground.
Using this setup I tried to pave the way to replace the teleoperator with an on-site handheld device.

Data Transfer

Multiplayer Connection

As mentioned above the approach to fullfil the data sharing requirement was to use an Unreal Engine multiplayer Connection.
There are different possibilites of online subsystems to use. As the plan is to use the handheld device on the construction site a local network connection was the choice for this project. Unreal Engine offers different types of multiplayer connections. Mainly two options could be relevant:

- Client - Server model
- Dedicated Server model

The first option allows both the server and the client to actively interact with the game. This is useful in case we want to perform any UE action on the excavator. In this case the excavator's UE instance would be the playing the part of the server and the handheld device would be the client.
The second option lets an Unreal Engine instance function as a real server without the possibility of interacting actively with the game. With this option the Unreal Engine instance on the excavator would serve as the dedicated server. In general this setup provides a more stable multiplayer connection however the possibility of interacting actively with the running UE instance would no longer be possible on the excavator.
In this project I used the Client - Server approach in case the necessity of input based interaction with the excavator UE instance arises in the future.

Connection Challenge

Normally when connecting two Unreal Engine games through a multiplayer connection it is the same game simply played from different devices. In this case however we have one UE instance running on the excavator using Linux both for development (ROS) and for the deployment (ROS Subscriber). The other game runs on Android and is an AR game. So in this setting we have a connection between two games which have fundamentally different components as well as different platforms that they run on.
It turns out that on the lowest level a successfull UE multiplayer connection requires identical checksums in the two systems. The checksum is a number generated from Unreal Engine which depends on the project name and certain components in the game level amongst other factors.
The AR components

State Sharing

To establish all data transfer connections between the two devices would have been too wide of a scope for this project which is why I focused on the transmission of the excavator's current state. In general if the transmission of any information succeeds using this setup then all further information can be packaged into game components accordingly and shared in the same fashion.
As seen in fig. 3.1 the excavator's state is fed to the Unreal Engine model directly through a ROS state subscriber node which is embedded in the game.
To solve the data transmission problem for the handheld device the idea was to intercept this connection. The state input is thus not only sent to the excavator model but also into a state storage game component. This game component would then be updated continuously and replicated in the handheld UE instance with the new values. Simultaneously if an excavator model is referenced by this state storage actor the state is fed into the model at each game tick to also update the handheld excavator model's state.

Colocalization

Bibliography

[1] D. Jud, S. Kerscher, M. Wermelinger, E. Jelavic, P. Egli, P. Leemann, G. Hottiger, and M. Hutter, “Heap - the autonomous walking excavator,” 2021.