Användarmanual

Hampus Westerberg
Joel Wiklund
Tomasz Mazurek
Carl Liljeberg
Mohammad Rajabi
Anton Lund
Simon Svahn

12 december 2024

Version 0.1

Autonom Truck

Status

Granskad		
Godkänd		

Projektidentitet

Grupp E-post: hamwe392@student.liu.se

Beställare: Gustav Zetterqvist, Linköpings universitet
Tfn: 013-28 19 94
E-post: gustav.zetterqvist@liu.se

Kund: Johan Lindell, Toyota Material Handling
E-post: johan.lindell@toyota-industries.eu

Handledare 1: Sebastian Karlsson (ISY)
E-post: sebastian.karlsson@liu.se

Handledare 2: Oskar Bergkvist (Toyota Material Handling)
E-post: oskar.bergkvist@toyota-industries.eu

Handledare 3: Andreas Bergström (Toyota Material Handling)
E-post: Andreas.Bergstrom.ext@toyota-industries.eu

Kursansvarig: Daniel Axehill, Linköpings universitet
Tfn: +4613284042
E-post: daniel.axehill@liu.se

Projektdeltagare

Namn	**Ansvar**	**E-post**
Hampus Westerberg	Projektledare (PL)	hamwe392@student.liu.se
Joel Wiklund	Testansvarig (TA)	joewi329@student.liu.se
Carl Liljeberg	Informationsansvarig (IA)	carli426@student.liu.se
Mohammad Rajabi	Dokumentansvarig (DOK)	mohra735@student.liu.se
Anton Lund	Mjukvaruansvarig (MA)	antlu106@student.liu.se
Simon Svahn	Designansvarig (DES)	simsv926@student.liu.se
Tomasz Mazurek	Sekreterare (SEK)	tomma870@student.liu.se

INNEHÅLL

DOKUMENTHISTORIK

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2024-12-04	Första utkastet	Simon Svahn	Alla

1 INLEDNING

Detta dokument är en beskrivning på hur man använder systemet från det färdiga projektet Autonom truck: Obstacle avoidance. Projektet gjordes i samarbete med Toyota Material Handling under hösten 2024 i kursen TSRT10 Reglerteknisk projektkurs. Målet med dokumentet är att ge all nödvändig information för att använda systemet som har utvecklats under projektet, men inte nödvändigtvis för att förstå hur det fungerar.

2 NÖDVÄNDIGA KOMPONENTER

2.1 Hårdvara

Den enda hårdvara som behövs är en dator. Användarmanualen är anpassad för en dator med operativsystemet Windows eller Mac OS. All simulering och beräkning tar mycket datorkraft, så en kraftfull dator är att rekommendera. Det finns dock inte några specifika minimikrav.

2.2 Mjukvara

Följande mjukvara behövs för att köra simuleringen.

2.2.1 *WSL 2*

Detta kapitel gäller endast om du använder Windows. WSL är Windows subsystem for Linux. Vissa har den redan installerat, annars följ instruktionerna här. Använd Ubuntu som Linux distribution.

2.2.2 *Docker desktop*

Docker desktop behöver installeras, detta kan göras här, (för Windows använd WSL 2). Docker är ett sätt att skapa en container vilken är ett isolerat operativsystem som installerar nödvändiga program och löser alla beroenden för att simuleringen ska fungera på datorn.

2.2.3 *Xlaunch*

För att kunna visualisera program när man är i en container så behövs en X-server, i vårt fall användes VcXsrv. Det går att installera här. Detta är dock bara aktuellt om man har en dator med operativsystemet Windows. Följande är viktigt när man kör VcXsrv:

- Display number ska vara -1
- Kör VcXsrv med "Multiple Windows"
- Använd "Disable access control"

2.2.4 *Visual Studio Code*

Som IDE används Visual Studio Code, detta kan installeras här. Två extensions behövs också:

- Remote Development Extension Pack
- Dev Containers Extension

2.2.5 *Git*

För att få tillgång till koden så måste Git installeras, det kan göras här. All kod för simulering finns i git-repot https://gitlab.liu.se/tsrt10/2024/toyota.

3 STARTPROCEDUR

3.1 Klona projektet

Kod för simulering kan klonas ner med följande kommando i terminalen:

```
git clone https://gitlab.liu.se/tsrt10/2024/toyota.git
```

3.2 Bygga bilden

Första gången projektet ska köras måste en docker-bild skapas. Starta programmet docker desktop. Se till att vara i projektets rootmapp och kör följande kommando i terminalen:

```
docker build -t minimal .
```

Detta kan ta en stund. Docker-bilden måste bara byggas om det finns nya program eller beroenden.
För att sedan starta Docker-containern, kör:

```
./run_win.bat
```

Har allt gått bra står det följande i din terminal:

```
root@docker-desktop:/ros_ws#
```

Detta gör att terminalen nu kör på dockerbilden (i princip en version av linux). Om man vill ansluta en annan terminal till samma bild kör man:

```
docker exec -it ros-env bash
```

3.3 GUI i webbläsare

Istället för att använda VcXsrv för att visualisera program kan man använda webbläsaren. Detta är nödvändigt om man har en dator med operativsystemet Mac OS.
Börja med att starta Docker Compose i en ny terminal:

```
docker compose up
```

Öppna VNC-anslutningen i din webbläsare. Gå till följande adress och tryck på "Connect" för att starta GUI:t:

```
http://localhost:8080/vnc.html
```

Anslut till containern via en annan ny terminal:

```
docker exec -it autonom-truck-2-ros-1 bash
```

Fortsätt med nästkommande instruktioner med samma terminal och de visualiserade programmen kommer dyka upp i webbläsaren.

3.4 Bygga projektet

Efter varje förändring i koden så måste projektet byggas om.

- Se till att Docker desktop har startat.
- Se till du är i Docker-containern, om inte kör:

```
./run_win.bat
```

För att bygga själva projektet kör:

```
colcon build
```

När bygget är klart, källkoda miljön genom att köra:

```
source install/setup.bash
```

3.5 Starta GUI

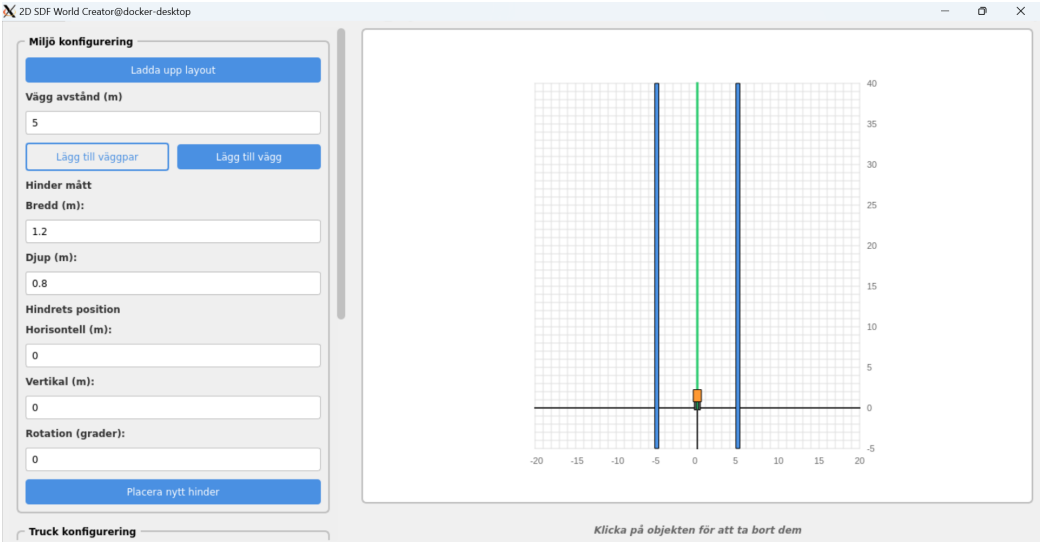
- Se till att Docker desktop har startat.
- Se till XLaunch eller GUI i webbläsare har startat.
- Se till du är i Docker-containern, om inte kör:

```
./run_win.bat
```

För att faktiskt starta systemet och dess GUI körs:

```
python3 src/GUI/world_define/world_gui.py
```

Användargränssnittet som visas på Figur 1 kommer då startas.



Figur 1: Användargränssnittet

4 GUI

Detta avsnitt förklarar hur gränssnittet används.

4.1 Bygga kartan

Kartan har ett koordinatsystem, trucken initialiseras vid koordinat $(0, 0)$. Truckens originalrutt är uppåt i kartan. Det går att lägga till väggar och hinder via menyn till vänster, och om man trycker på föremål på kartan så tas de bort. Se Figur 1.

4.1.1 *Vägg*

Det går att sätta ut väggar i kartan. I en textruta kan det anges vilket avstånd från centrumlinjen väggen ska placeras. Genom att trycka på `"Lägg till vägg"` så läggs väggen till i kartan.

4.1.2 *Hinder*

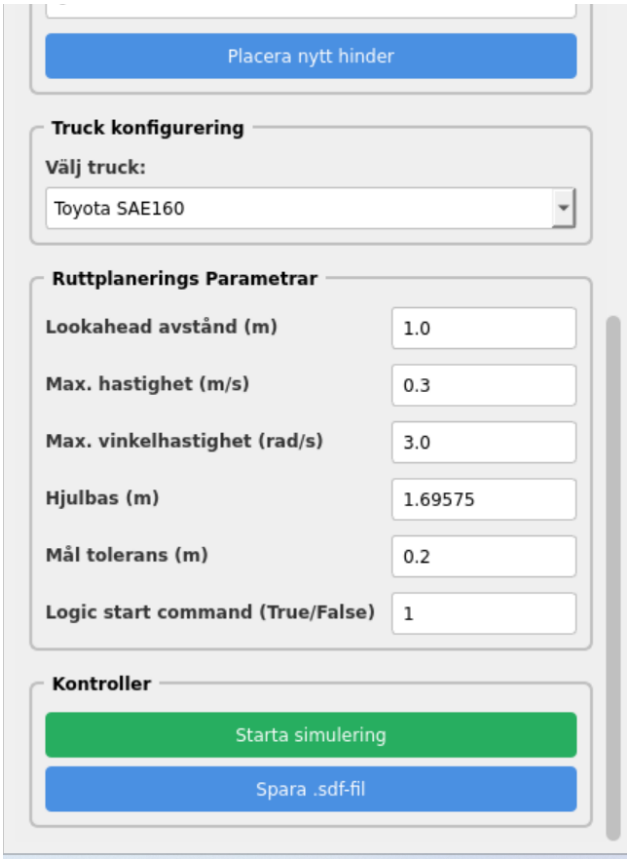
För att lägga till hinder anges hindrets höjd och bredd i meter. Dessutom anges hinders koordinater vertikalt och horisontellt samt hindrets rotation i grader. Genom att trycka på `"Lägg till hinder"` så läggs hinder till i kartan.

4.1.3 *Konfigurering*

I stället för att skapa en ny karta kan man ladda in en redan färdig karta som en .sdf-fil. Det görs genom att trycka på `"Ladda upp layout"`. Det går även att spara sin karta som .sdf-fil under `"Spara .sdf-fil"`. Dessutom går det att välja mellan fördefinierade truck-modeller att använda använda under `"Truck konfigurering"`. Se Figur 2.

4.1.4 *Ruttplanerings Parametrar*

Ett antal parametrar för ruttplaneringen kan bestämmas i GUI:t. Den första är "Lookahead avstånd" det är en parameter för pure pursuit, vilket är en ruttföljningsmetod. En annan parameter "Logic start command" sätts till 0 om man vill att trucken ska köra direkt, den sätts till 1 om man vill bestämma själv när man vill att trucken ska köra. De andra parametrarna är självförklarande.



Figur 2: Användargränssnittet konfigurering

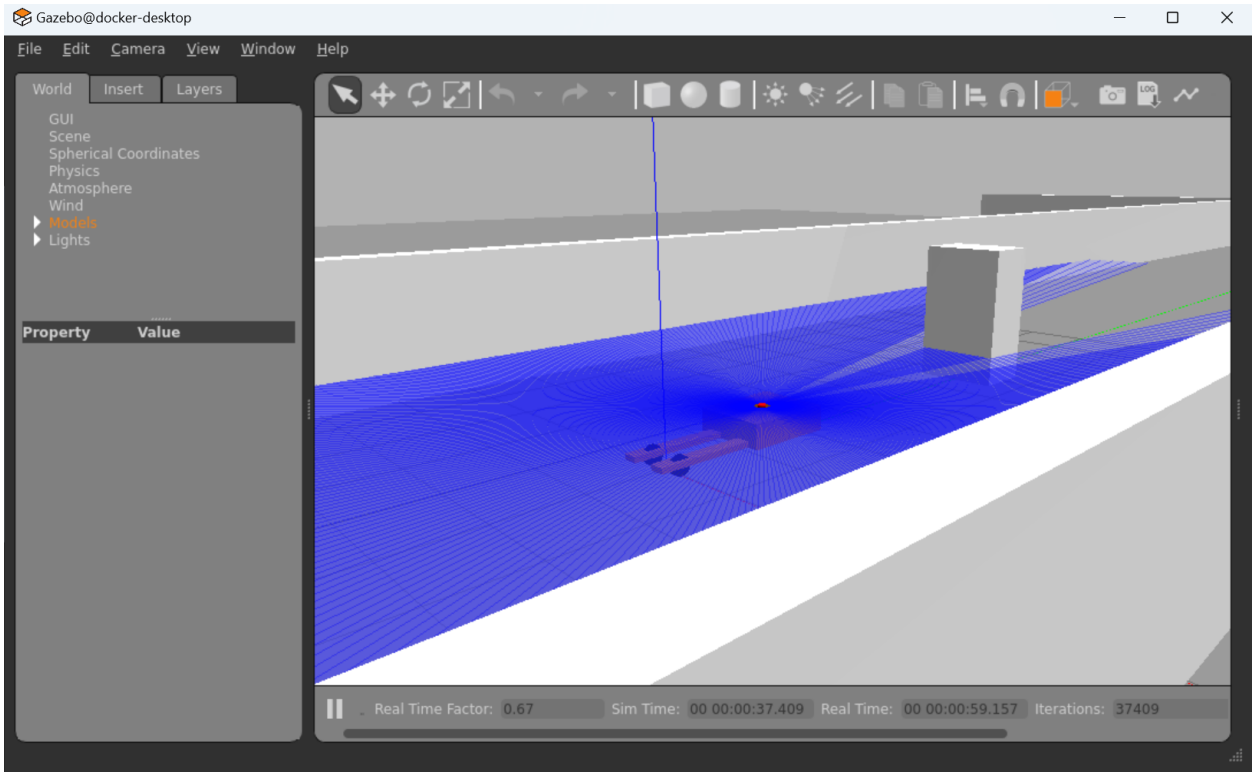
5 SIMULERINGEN

För att starta simuleringen ska du trycka på "starta simuleringen" i GUI:et. Då kommer några olika fönster att öppnas.

5.1 Gazebo

Här körs själva simuleringen. Simuleringen sker i 3D med hänsyn till truckens kinematik. Även sensordata från truckens LIDAR och odometer simuleras här för att användas. Allt visualiseras i Gazebo-fönstret, Se figur 3. Gazebo tar

mycket datorkapacitet, speciellt för att visualisera, det går att köra Gazebo med bara simulering utan visualisering, det gör man genom att kryssa i rutan för det i GUI:t.



Figur 3: Gazebo

5.2 RViz

I RViz visualiseras vad trucken ser och vad den vill göra. Kartan är uppdelad i ett rutnät. Till vänster finns en panel där man kan bestämma vilka saker man vill och inte vill visualisera. Se figur 4. Informationen som visas som standard är:

VITA RUTOR: Områden som trucken tror är fritt

SVARTA RUTOR: Områden som trucken tror är ett hinder.

GRÅA RUTOR: Områden som trucken ännu inte känner till.

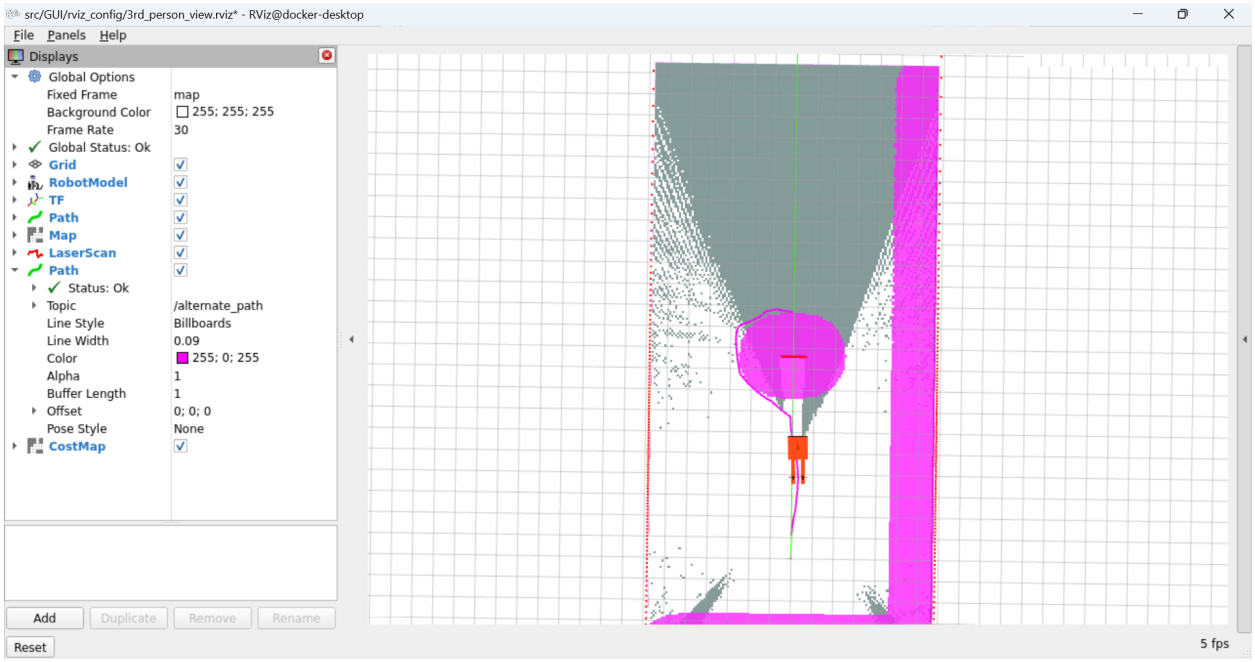
GRÖNA LINJEN: Truckens originalrutten.

ROSA LINJE: Indikerar truckens omplanderade rutt som, när sådan finns.

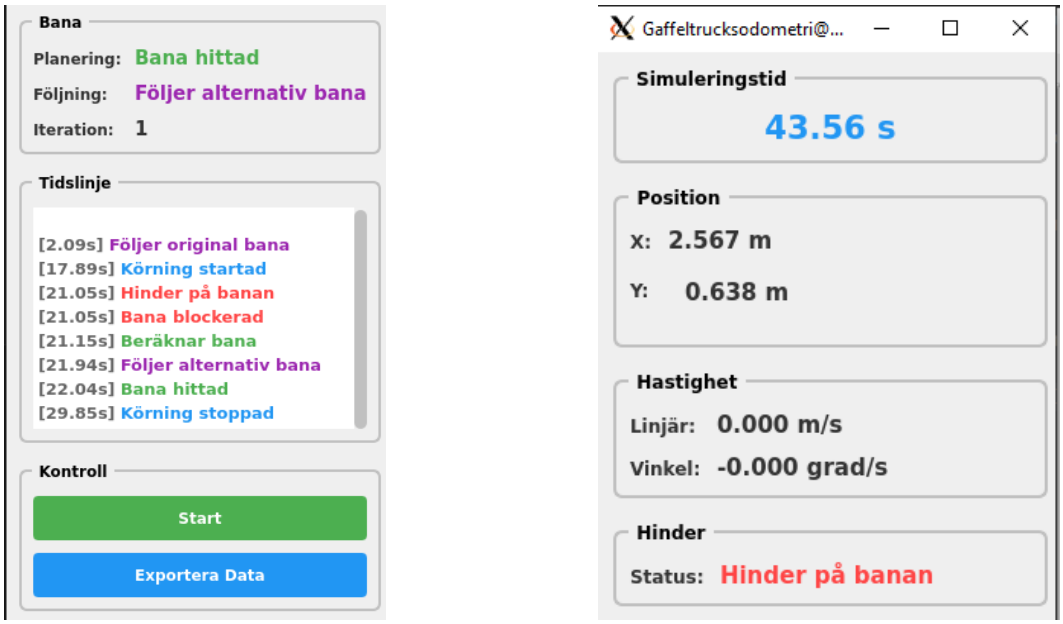
ROSA OMRÅDET: Indikerar det expanderande området som ser till att trucken inte kör in i något.

RÖDA PRICKAR: Träffar från den simulerade LIDAR-enheten.

BLÅ LINJE: Den faktiska rutten som trucken har kört.



Figur 4: RViz



Figur 5: Kontrollpanel

5.3 Kontrollpanel

Kontrollpanelen skriver ut viktig information under körningen samt möjliggör att starta och pausa körningen. Se figur 5. De olika rutorna är:

SIMULERINGSTID: Skriver ut hur länge simuleringen har pågått.

POSITION: Skriver ut truckens x och y-koordinater.

HASTIGHET: Skriver ut truckens fart och rotationshastighet.

HINDER: Skriver ut om den har hittat ett hinder.

BANA: Status på rutten, om den är blockerad eller inte.

TIDSLINJE: Skriver ut händelser för trucken och vid vilken tidpunkt.

KONTROLL: Ger möjlighet att Starta/Pausa och exportera data för körningen.

5.4 Nerstängning

För att stänga programmet bör du klicka Ctrl+C i terminalen där GUI:t startades. Detta stänger alla fönster på ett korrekt sätt.