POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Multi-sensors SLAM simulation for Planetary Rover Exploration

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: Davide Viviani

Advisor: Prof. Mauro Massari

Academic year: 2021-2022

---

1. Introduction

Nowadays, robotic planetary exploration is one of the top researched area in the field of space exploration. Through the last decades several rovers have been deployed on the surfaces of Moon and Mars, with more and more advanced technologies. However, in particular for the case of Mars, the delay in the communications with the Earth has brought complications to their mission planning and has reduced the possible range of tasks achievable. Currently, space agencies from all over the world are pushing to build more autonomous and reliable systems for rover exploration.

In this current state, a particular focus has been brought to the problem of exploring a planetary surface and at the same time making the agent capable of localizing itself in this new environment. This is problem is called SLAM problem for Planetary Exploration and it is going to be the focus of this work.

The thesis here presented aims at creating a realistic and reliable simulation environment to test and collect data for a planetary rover SLAM algorithm. The first section is going to introduce the problem, then the simulation environment and the SLAM test are analyzed, presenting the results and possible future developments in the final section.

2. Simulation environment

The first problem tackled has been the creation of a simulation environment capable of producing realistic, truthful and reliable results. The purpose of reproducing such a virtual environment is to have available a tool which allows to test the SLAM algorithm before its hardware implementation. Moreover, the virtual environment can be set to reproduce real areas to be explored, maybe exploiting data from satellite images or previous rover missions. The simulation environment has been built with the cooperation of four different softwares: Blender, Unreal Engine, Simulink and Matlab.

2.1. Mars virtual surface

The first step has been to recreate a surface similar to a portion of the Martian soil and it has been achieved by means of Blender. Blender has proved to be a very powerful software when reproducing detailed surfaces and textures. Figure 1 shows the final virtual Mars surface recreated. The surface presents typical Mars features like the dusty soil texture, the dark volcanic rocks, the reddish light and the dust covering all the surfaces. These are important details which are going to affect some of the choices adopted for the SLAM algorithm in the next sections.

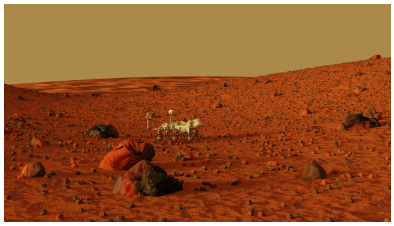

Figure 1: Mars virtual surface

2.2. Rover trajectory simulation

Once an enough realistic and detailed surface has been recreated with Blender, it has been exported in an Unreal Engine scenario: this choice has been done because of the powerful capabilities of Unreal Engine in the field of movement simulation. Unreal Engine has been coupled with a Simulink file, allowing the setup of the various parameters to be easier and more immediate by means of a Matlab script. In this way the desired trajectory for the rover has been drawn manually over the map of the environment and subsequently the simulation has been activated both in Simulink and Unreal Engine simultaneously.
This procedure is based on the possibility to link Unreal Engine and Simulink by means of the *Automated Driving Toolbox*. Figure 2 shows the nominal trajectory drawn in Matlab and the Simulink architecture for the simulation.

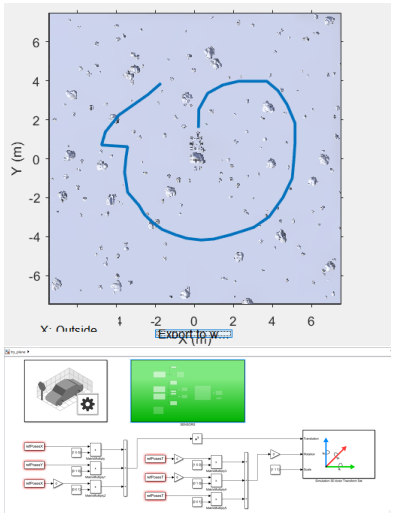

Figure 2: Rover trajectory and Simulink architecture to reproduce it in Unreal Engine

2.3. Sensors simulation

It has been explained how the rover trajectory is simulated. The simulation tool build allows also to reproduce realistic sensor data by means of the shown architecture. Indeed, this simulation has been built over a rover equipped with both a Lidar sensor and a StereoCam sensor. Again, the reproduction of the sensor has been done linking the Simulink file to Unreal Engine. The powerful architecture proposed has the capability of both making the data acquisition be followed live during the simulation and also storing all the sensors acquisition in Matlab variable. In particular, this last feature is going to be fundamental through the next steps of the SLAM algorithm. Figure 3 shows the visualization of the environment from the StereoCam while the simulation itself was running.

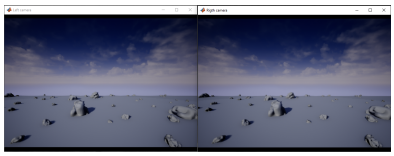

Figure 3: StereoCam live view

2.4. Data elaboration

As in the real case, the data from the sensors come in the form of raw information which have to be elaborated in order to make it useful for the following SLAM algorithm. Both the Lidar data and the StereoCam data have been rearranged in the form of point clouds inside which clusters of point have been identified as landmarks. It should be pointed out that this in not the unique form in which the sensor data could reveal themselves as useful, however in this work this form has been adopted in order to clearly define geometrical landmarks and use the same Data Association algorithm for both the sensors. The point clouds from the sensors have been scanned in order to divide the point inside them into clusters, where each cluster is identified as landmark. Each cluster in made by point under a selected Euclidean distance and over a certain Angular distance one from the other. By means of this clustering procedure, several landmarks have been identified inside each of the sensor acquisition at every time instant t of the simulation. Then, for each of the clusters, its geometric centre has been computed and a bearing and range distance in the rover local frame have been assigned to each landmark. Figure 4 shows an example of a clustered point cloud (in this case from a Lidar acquisition).

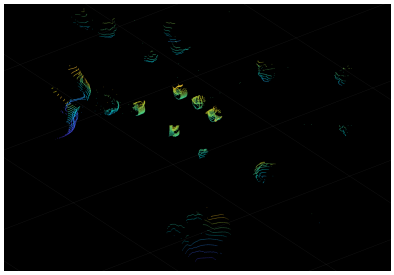

Figure 4: Clustered point cloud

The simulation architecture of the first part can be summarized as in the following algorithm:

Algorithm 1 Simulation Part 1

Require: Unreal Engine scene set, Simulink model set
1: Open Matlab
2: Run the main script and draw the trajectory
3: Smooth the trajectory and generate control input u
4: Open Simulink and set simulation time
5: From Simulink open Unreal Engine
6: In Unreal Engine open the simulation environment
7: Run the simulation first in Simulink, then in Unreal Engine
8: **return** u, *StereoCam data*, *Lidar data*

The point cloud elaboration and clustering procedure is instead shown here:

Algorithm 2 Simulation Part 2

Require: *StereoCam data*, *Lidar data*
1: Upload the Lidar and StereoCam data
2: **for** Lidar data, StereoCam data **do**
3: Transform the data into one point cloud for each time step
4: **for** each point cloud **do**
5: Cluster the points inside the point clouds for distance and angular separation
6: Count the clusters (or landmarks)
7: Compute clusters geometric centre coordinates
8: Compute geometric centre orientation in the rover local frame
9: **end for**
10: **end for**
11: **return** Lidar Point Cloud and landmarks, StereoCam Point Cloud and landmarks

3. SLAM formulation and solution

The SLAM problem is analyzed in this work under its probabilistic formulation. The Simultaneous Localization and Mapping procedure is established as a statistical basis for describing relationships between landmarks and uncertainty [2]. It has been shown that an high degree of correlations between the estimates of landmarks locations in the map is present and these correlations are growing with time as the number of observations increases. The correlation between the landmarks estimates is present because of the uncertainty in the estimate of the vehicle location, which propagates into the observations [3]. Some quantities should be introduced:

- x_k: state vector containing location and orientation of the robot;
- u_k: vector containing the control inputs at different times;
- m_i: vector with the location of the landmark i;
- z_{ik}: observation of a landmark i at time k, written also as z_k.

Also the following sets have to be defined:

- $X_{0:k} = [x_0, x_1, ..., x_k]$: history of vehicle locations;
- $U_{0:k} = [u_1, u_2, ..., u_k]$: history of control inputs;
- $m = [m_1, m_2, ..., m_n]$: the set of all landmarks;
- $Z_{0:k} = [z_1, z_2, ..., z_k]$: the set of landmarks observations.

The probabilistic SLAM requires the following probabilistic distribution to be computed at all times k:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \tag{1}$$

This probability distribution describes the joint posterior density of the landmark locations and vehicle state at time k given the observations and the control inputs up to and including time k, together with the initial state of the vehicle. This formulation is exposed in details in [2].

Following this formulation and the application of the Bayes theorem, the probabilistic SLAM can be reformulated in a recursive two steps version, based on a *motion model*, representing how the quantities inside the algorithm are updated after the control input is applied, and a *measurement model*, representing how the same quantities are updated after the sensor measurements. This recursive prediction-correction implementation form: the SLAM problem can be reformulated in two steps, allowing a recursive prediction-correction implementation form.
Time-update:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \\ = \int P(x_k \mid x_{k-1}, u_k) \\ P(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1} \quad (2)$$

Measurement-update:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \\ = \frac{P(z_k \mid x_k, m) P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0)}{P(z_k \mid Z_{0:k-1}, U_{0:k}} \quad (3)$$

Equations 2 and 3 provide a recursive procedure for calculating the joint posterior in equation 1. Once the probabilistic formulation of the problem has been introduced, it is possible to implement its solution. The solution to the probabilistic SLAM presented in this work is an Extended Kalman filter formulation.

3.1. Extended Kalman filter SLAM

The Extended Kalman filter SLAM has been implemented following algorithm 3:

Algorithm 3 Simulation Part 3

Require: *StereoCam point clouds*, *Lidar point clouds*, u
1: Upload Lidar or StereoCam point clouds
2: **for** each time step **do**
3: Actuate the control input
4: Run the EKF Prediction step
5: Run the Data Association
6: Run the EKF Correction step
7: **end for**
8: **return** Results and plots

Step 4 and 5 of algorithm 3 are implemented in a classical formulation of the Extended Kalman filter, while particular focus during the work has been adopted over step 3: the Data Association problem.

3.2. The Data Association problem

Particular attention has been adopted when tackling the Data Association problem. As the simulation aims at reproducing real environments, it also brings with itself the consequent real Data Association issues. In particular, in the case of planetary SLAM under study, it is important to find a way to recognize and associate correctly different rocks seen during the rover exploration. The Data Association problem is here formulated following the work in [1], in particular:

Algorithm 4 Data Association procedure

Require: point cloud at time t, point cloud at time $t-1$ (both with landmarks defined by algorithm 2)
1: Filter out the rover parts from the point clouds
2: Huang-Arun algorithm finds the movement between the point clouds as a rotation matrix R and a translation vector $\vec{t}$
3: Match the features in the point clouds to select the observed landmarks
4: Associate to each observed landmark a range and a bearing measurement
5: Select only the best associations
6: **return** Point cloud at time t with only its best associated landmark

The procedure used in this thesis is a basic geometrical procedure for the Data Association, due to fact that the association problem is not the main focus of the research. However, it enables effective associations and the Extended Kalman filter algorithm to work properly.

4. Results

This section presents the result of the EKF SLAM simulation inside the virtual environment built in section 2. The results of the algorithm are analyzed together with the variations of some of its parameters and under the important assumption of the absence of any Loop Closure. This assumption stands for the fact that the recognition of already visited places when the rover revisits previously seen areas in not implemented in this thesis.

Figure 5 shows the comparison between the desired trajectory, a trajectory affected by random noise and the same corrected by the EKF SLAM.

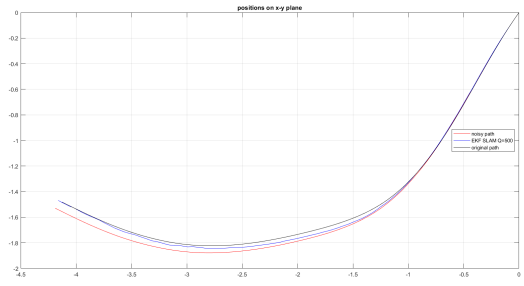

Figure 5: EKF SLAM results

In the figure the black trajectory represents the nominal desired one, the red trajectory is the noisy one and the blue one is the trajectory under the effects of the EKF SLAM. As expected the blue trajectory oscillates around the nominal one, with an error behaviour represented here in figure 6.

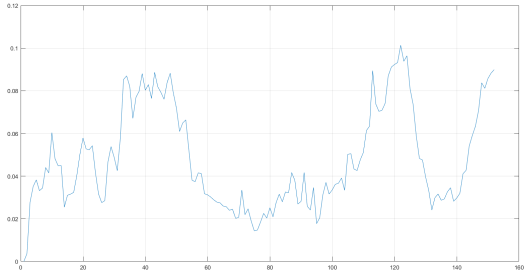

Figure 6: EKF SLAM error

As can be noted the error oscillates, however it slowly increases. This phenomenon is demonstrated to be an effect of the absence of Loop Closures: even if the data association algorithm and the EKF are able to correct the trajectory, after a certain distance covered by the rover, the error becomes too big and impossible to reduce again to a zero value.

4.1. EKF SLAM behaviour with parameters changes

This section analyzes the changes in the results of the EKF SLAM when changing the values of its parameters. In particular, the results are analyzed for changes in the sensor uncertainty matrix Q and for changes in the trajectory shape. Figure 7 shows how the SLAM behaviour changes with the uncertainty on the sensor acquisitions and consequently on the relative power of the correction step in updating the state vector. Higher values for the terms of Q

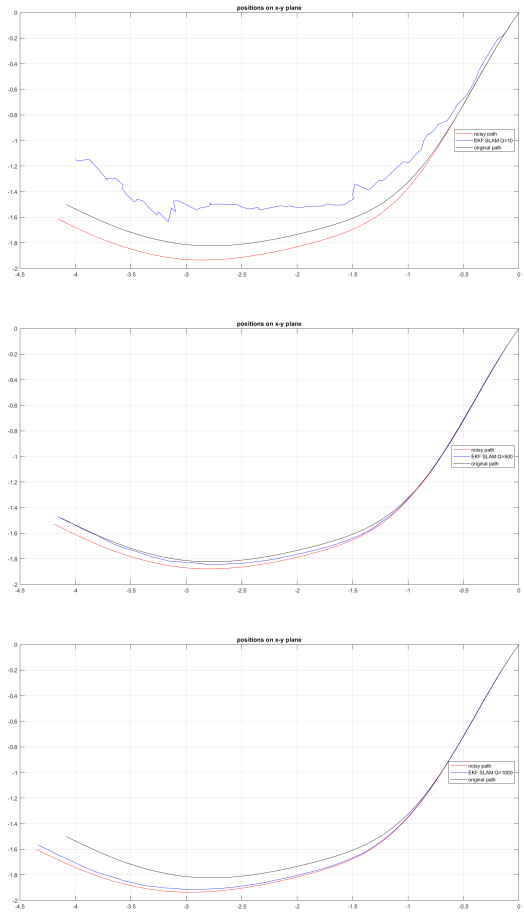

Figure 7: Different Q comparison

bring a less powerful correction step making the SLAM incapable to move enough towards the nominal trajectory. At the same lower values bring less smooth corrections, often bringing to drastic corrections and increasing errors. A fine tuning of Q must be performed.
The behaviour of the EKF SLAM is then analyzed with respect to the shape of the trajectory. The results show how the algorithm has a faster rate of errors accumulation when the trajectory is curved. The algorithm showed a linear progression of the error when moving on straight paths, while a quadratic rate showed for curved path. Again, this result was expected in the absence of loop closures and it is confirmed by the literature. Both the simulation and the SLAM algorithm have been so proved as effective and the results confirmed the expected behaviours.

5. Conclusions

The main outcome of this simulation is the possibility to recreate real problems of a planetary SLAM in a virtual environment. Thanks to this result, it is possible to study and analyzing the algorithm behaviour before its hardware implementation and so correct what needed before the rover construction, allowing a more economic, both in time and money, construction approach to planetary rovers.
However, the work is far from complete and several future developments of this work are shown here:

- *Data Association*: the problem of a more powerful data association can be tackled, improving the results with new techniques or exploiting information as, for example, the light reflected by the observed surfaces;

- *In Loop Simulation*: the simulation can be structured as a single loop process, without the partition in three parts, where the rover pose is fed to Unreal Engine and the sensor data are fed directly to the SLAM in a closed loop;

- *Simulation scaling*: the architecture proposed for the simulation can be exploited to other simulations, scaling the Simulink-Unreal Engine link to more complex and articulate simulations.

The developments above are only part of the possible improvements fro the simulation architecture proposed.
In conclusion, the desired outcomes from this thesis work have been achieved and can be divided in three main areas:

1. *Simulation environment*: by means of the cooperation between the software of Blender, Unreal Engine and Simulink, it has been possible to reproduce a detailed Mars surface and also to simulate realistic sensor acquisitions;

2. *Sensor data elaboration*: both in the case of the Lidar sensor and the StereoCam sensor it has been possible to recreate the process of elaboration of the sensor data from the raw acquisition to a more refined version to be fed to the SLAM algorithm;

3. *SLAM algorithm*: both the realistic simulation environment and the sensor data elaboration allowed to simulate an Extended Kalman filter SLAM with the problems of a real SLAM: the imperfections in the sensor acquisitions and in the data association brought the SLAM to a realistic level and implementation in which the final results were satisfying.

With the result explained this thesis work is considered concluded and ready for the implementation of new powerful features.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9, 1987.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32, 2016.

[3] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.