Ukrainian Catholic University

Faculty of Applied Sciences

Computer Science Programme

Simultaneous Localization and Mapping

Linear Algebra final project report

Authors:
Augusta Ada King-Noel Lovelace
Charles Babbage

May 2022

Abstract

Bla bla about the importance of SLAM. (robotics and so on). We implemented the proposed algorithm by means of pyopencv. Source code: [1].

1 Introduction

- what is SLAM (visual slam) - different approached (Monocular, Stereo, RGBD and so on), just mention, could be described below. - frontend / backend - we focused on frontend (get the ddiagram from the book).

2 Problem setting

2.1 Problem formulation

With the given setting, we are interested mainly in two parts:

1. Modeling the object's motion. In an abstract setting this can be modeled as $x_k = f(x_{k-1}, u_k, w_k)$, where x_k denotes a current position of an object, x_{k-1} - its previous position, u_k - change of its state and w_k - noise.

2. Collecting the landmarks of a nearby environment. The observed data $z_{k,j}$ will then be described by this equation: $z_{k,j} = f(y_j, x_k, v_{k,j})$. Here y_j - landmark point, x_k - its coordinates, $v_{k,j}$ - noise of the observation.

These two equation summarize the SLAM. For the Euclidean space the motion model can be expressed as the so-called special Euclidean Group:

$$SE(3) = \{T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in \mathbf{R}^{4x4}\},$$

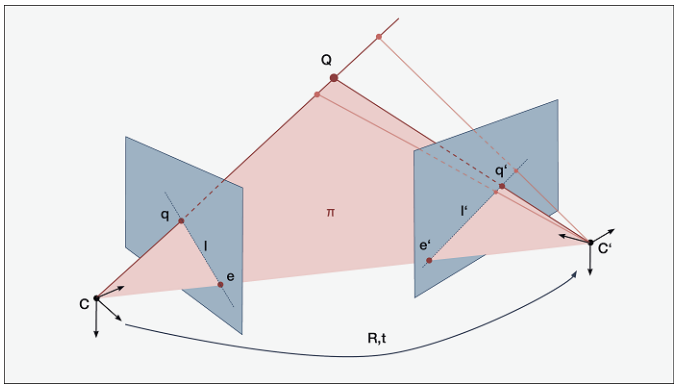
where R - rotation matrix, t - translation vector. It allows to express state a_1 as

$$a_1 = Ra_0 + t.$$

Important note here is that we associate the state of an object with the location of camera that produces some input data.

2.2 Epipolar geometry

The object motion can be found from the correspondence between two consecutive video frames.



According to the camera model, each of the landmarks Q will be projected onto 2-D plane of pixels as q, q'. Motion, described by the rotation matrix R and transformation vector t can be reproduced than via Epipolar constraint.

Pixel points q, q' are described by the following equations:

$$\begin{cases} s_1 p_1 = KP \\ s_2 p_2 = K(RP + t) \end{cases}. \tag{1}$$

As was discussed in the setting, due to the limitations of the approach, we can represent the points only to some constant factor. Moreover, it is worth mentioning that p_1, p_2 will be the points matched by the feature matcher.

To find the coordinates on the normalized plane of two pixels, we can simply take the inverse of the matrix representing a camera: $x_{1,2} = K^{-1} p_{1,2}$. Then the equation above can be rewritten as $x_2 = Rx_1 + t$. This can be further transformed into

$$p_2^T K^{-T} t \times R K^{-1} p_1 = 0.$$

This equation is referred to as epipolar constraint. For the sake of simplicity, the notions of Fundamental and Essential matrices are introduced:

$$\begin{cases} E = t \times R \\ F = K^{-T} t \times R K^{-1} \\ x_2^T E x_1 = 0 \\ p_2^T F p_1 = 0 \end{cases}. \tag{2}$$

It is clear that usage of Essential matrix will make the calculations more accurate, as it uses precomputed calibration parameters for the camera. The process of camera pose estimation can then be summarized as:

1. Find E or F from the known pixel positions.

2. Find R, t based on those matrices.

In practice, the Essential matrix is found via Eight-Point algorithm and then R, t are found via SVD and triangulation to get the result that would satisfy the constraints given by the known projections of landmarks.

$$\begin{cases} R = U R_{Z(\pm pi/2)} V^T \\ t = U R_{Z(\pm pi/2)} \Sigma U^T \end{cases}. \tag{3}$$

2.3 Triangulation

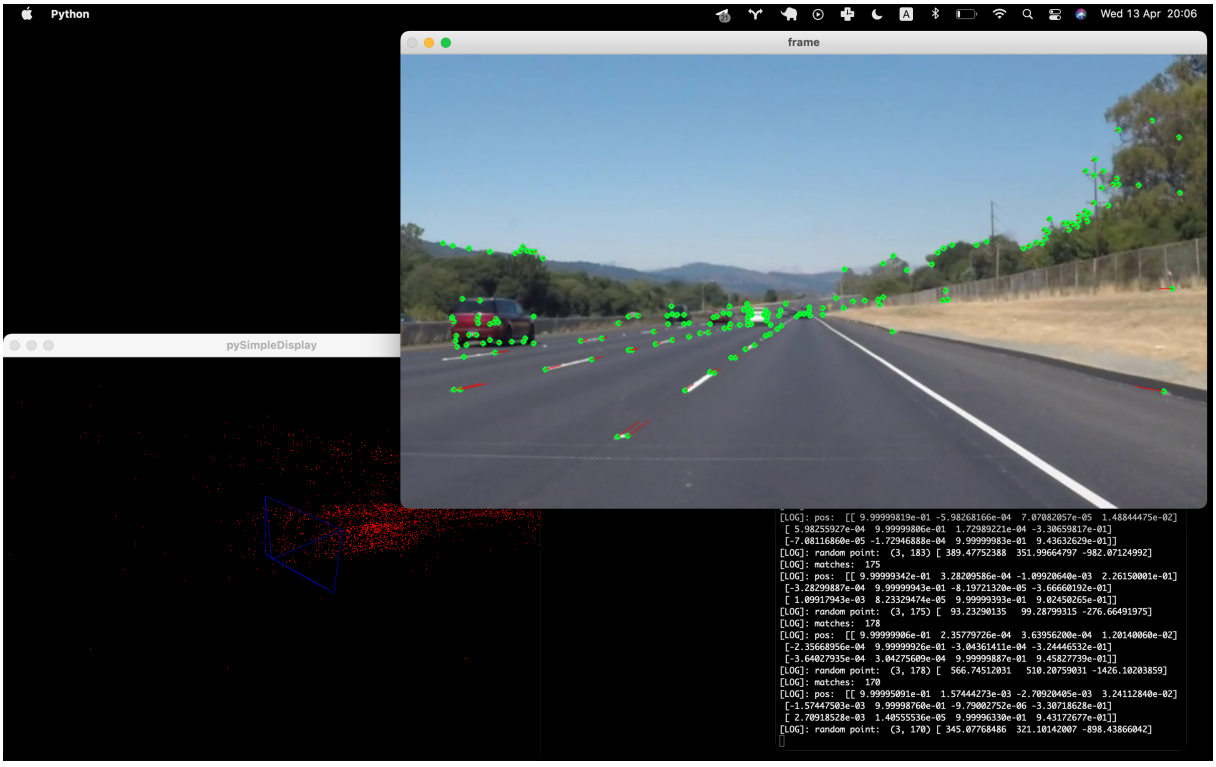
3 Overview of approaches

4 Implementation

Data: stream of video frames
initialization;
calibrate camera;
while *true* **do**
    get frame;
    get the features;
    match the features between current and previous frame;
    find essential matrix;
    find rotation matrix and transformation;
    store the movement and point map;
    update previous frame;
end

Algorithm 1: General algorithm of SLAM

The algorithm was implemented in Python via pyopencv. The visualizer of the SLAM was written via OpegGL. For the feature extraction the ORB was used due to its computational efficiency. Feature matching was done in brute-force manner because limitation on the number of features was only several hundreds. Fundamental matrix was then constructed as we lack the camera parameters for the test video.



5 Experiments

6 Limitations

There are two main limitations of such approach:

1. Triangulation is only possible if the enough amount of translation is present. If the amount of translation is small, pixel's change would result in big depth uncertainty. On the other hand, if translation is too big, the algorithm may fail to match the features. This problem will be especially seen in cases of so-called pure rotation, when there is no translation. Though in not so extreme cases, there are means to make this problem less visible - delayed triangulation etc.

2. Assumption about static sight of view brings a lot of limitations as well. For instance, the algorithm cannot be applied to autopilots as the moving object would make it think that the camera has moved.

7 Conclusions

References

[1] `https://github.com/bohdanhlovatskyi/OhISee`