Software requirements specification for project iSLAM (indoor-SLAM)

1. Authors

Kozlov Kirill, Shalygin Igor, Tsoy Anton

2. Introduction

We make iSLAM. The goal of our project is to develop software that constructs a 3D map of the room based on the video from a mono camera. This goal can be reached using Indoor SLAM. The general algorithm of our solution is described here:

1) Via terminal the user executes our program passing a specified video source, file with calibration data and path to the output file for 3D map.

2) Main function process passed video frame-by-frame using monocular SLAM with:

- FAST feature extractor
- SSD/KLT Feature tracking
- Linear algebra methods (at first, SVD) for estimating changes between frames
- Linear triangulation with placing points to world coordinates

3) Constructed 3D map is saved in a specified directory and can be used for any purpose.

3. Glossary

Software refers to a collection of programs, procedures, algorithms, and its documentation that allows for the functionality of a computing device or system.

Program - Instructions that a computer can interpret to perform a specific task.

SLAM (simultaneous localization and mapping) is a method of construction of the area map by processing video captured by a moving camera.

Indoor SLAM is a specialized application of the SLAM, designed specifically for indoor environments.

Monocular SLAM is a SLAM using a **mono camera** for capturing video.

A 3D map is a spatial representation of an environment that is generated using three-dimensional data. This map provides a detailed and accurate visualization of an area, offering information about the height, width, and depth of the objects within it.

Mono camera is a single lens camera.

Video Source refers to the origin of the video footage that will be used for the 3D map construction.

Calibration - The adjustment of device parameters to ensure accurate output, often used in SLAM to align and synchronize sensors.

Calibration matrix - matrix with camera's calibration.

Feature refers to a distinct and recognizable element within the environment that the algorithm can identify and track.

Clear frame is a frame with enough feature's count. (Concrete lowest bound will be estimated later on practice).

Rotation matrix is a 3x3 orthogonal matrix for rotating point on some angle due to matrix multiplication.

Translation vector is a simple vector with 3 values for translating 3D point due to addition.

Projection matrix is a matrix 3x4 where columns 1-3 is a 3D rotation matrix and the last column is a transposed translation vector. This matrix is used for changing 3D column point (rotation and translation at once operation): $p_{moved} = P \cdot p_{old}$

World projection matrix is a projection matrix for placing points relating to current camera position.

World camera pose is a simple 3D point which presents the current camera position.

Origin projection matrix is a projection matrix with identical transformation: $[I_{3,3}|(0,0,0)^T]$

Epipoplar line is a projective geometry abstraction. These lines pass through camera centers of 2 images.

Essential matrix is a 3x3 matrix which relates corresponding points in two frames from a calibrated camera using epipolar lines. This matrix is used for extracting possible rotations and translations between these frames.

SVD (Singular values decomposition) is a decomposition of a given matrix onto 2 orthogonal matrices (U, V) and 1 diagonal matrix (D). $A = UDV^T$. Values from these matrices have a wide application in CV.

Linear triangulation is a method for getting 3D homogeneous points using 2 corresponding points arrays and projections matrices with the help of SVD

Homogeneous point is a point with additional parameter for convenient calculation with proportions saving. This construct allows to show points at infinity (additional parameter is 0).

To track features - it means to match the feature from the previous frame and the feature in the current frame.

Frames batch is a finite set of frames.

Output directory is a directory where constructed map files are located.

4. Actors

a. **User** is a person or other program which wants to get a 3D map for any purpose

5. Functional requirements

5.1.1. Use-case "3D map construction"

Actors: user.

Goals: to construct a 3D indoor map of some location.

Precondition: user has video source with calibration for it.

Main success scenario:

1) User launches a program from the command line with arguments for video source, calibration matrix and output directory.

2) Program calls main video processing cycle with static parameters which will be described below:

1. Program initializes world projection matrix and origin projection matrix as $[I_{3,3}|(0,0,0)^T]$ and initial camera position as $(0,0,0)^T$ and the processing video cycle starts.

   Cycle iteration description:

2. Program finds the first clear frame using FAST feature extractor by checking `REQUIRED_FEATURES_COUNT` static parameter and saves it as previous frame for further processing
3. Program collects frames batch with size `BATCH_SIZE` static parameter
4. Program tracks previous taken features to features at frame from batch and skips frames where was less than `REQUIRED_FEATURES_COUNT` tracked features count.
   a. Here we try to find the best frame in the batch.
   b. If all frames are bad, we skip the batch and return to step 2.
5. Program estimates the projection matrix between extracted points `q` from the previous frame and related 1-by-1 tracked points `g`.
   a. Finds essential matrix using calibration matrix and arrays `q`, `g`
   b. Construct possible rotation matrices R_1, R_2 and translation vectors t, -t using essential matrix singular values decomposition
   c. Chooses best combination from $[R_1, t]$, $[R_1, -t]$, $[R_2, t]$ and $[R_2, -t]$ by counting points at the front of cameras (positive Z coordinate) using linear triangulation with other projection as origin projection. Best combination is the new projection matrix.
6. Program constructs new homogeneous 3D points in local coordinates using triangulation (method may differs from method in step 5.c) with `q`, `g`, origin projection and projection from step 5
7. Program normalizes homogeneous points by dividing coordinates on W (the fourth parameter of homogeneous point)
8. Program projects 3D points to world system by multiplying with world projection matrix

9. Updates world projection matrix and world camera pose by multiplying with projection matrix from step 5
10. Writes new camera position to `<output_dir>/pose.txt` and world 3D points to `<output_dir>/3Dpoints.txt`
    a. The points in the files have format:

       `<X coordinate>, <Y coordinate>, <Z coordinate>;`

       Points from each frames' pair are written in separate `[ ]`
11. Update data for next iteration:
    a. Current frame extracted points vectors are moved to previous frame extracted points vectors
    b. Skipped frames are deleted from the batch.

Alternative scenario "*incorrect paths*":

Trigger condition: some input file's path or output file's directory doesn't exist.

1) Execution interrupts and the program asks the user to specify correct paths in arguments.

Alternative scenario "*external interruption*":

Trigger condition: any external event interrupts the program.

1) Undefined part of the map was processed.

2) The processed points are saved to the file 3Dpoints.txt in the output directory.

5.1.2. Use-case "Calibration"

Actors: user.

Goals: to get calibration of video source.

Precondition: user has a video or numbered photos from the same camera with chessboard or its dummy.

Main success scenario:

1) The user runs the program from the command line with a calibration flag, specified path to video or directory with photos and path to directory where calibration data will be saved.

2) Program detects chessboard corners in each photo or in frames with a small time gap and saves them.

3) Program composes all detected corners arrays with real chess board sizes (which are known) and computes intrinsic camera parameters

4) Program saves parameters to .xml file in directory specified by user

Alternative scenario "*invalid paths*":

Trigger condition: video or directory specified by user doesn't exist.

1) Execution interrupts and the program asks user specify correct path

Alternative scenario "*not enough data for calibration*":

Trigger condition: there're not enough photos with the chessboard or clear frames with it in video.

1) Program notifies user about bad source for calibration and terminates

5.1.3. Use-case "Visualization"

Actors: user.

Goals: to see a constructed 3D map of points.

Precondition: user has a .txt file with points got from our program.

Main success scenario:

1) The user runs the program from the command line with the path to the file with 3D points as an argument.

2) A GUI window opens on the user's screen where the 3D points from the file will be visualized.

3) The utility reads coordinates of points from the file and visualizes them in the GUI window one by one.

4) Once all points have been read and visualized, they will be displayed in the GUI window until the user closes it.

Alternative scenario "*invalid contents of input file*":

Trigger condition: the user has submitted a file other than the file that was received as a result of our program, or the user has corrupted a file that was received by our program.

1) Execution interrupts and the utility prompts the user to specify file path with the correct contents.

*The user can enter another data file with the correct format, but the points that will be visualized will not match those processed by our main program.

7. Non-functional requirements

7.1. Environment

The program is supported only for x64 Windows 10/11.

7.2. Performance

Using hardware:

- Ryzen 5 5600h
- Geforce RTX 3060 130 watt.
- 16 GB RAM
- SSD 1 TB.
- SSD 512 GB.

The video with a duration of 3 minutes and resolution 1280 x 720 was processed in 40 minutes using the SSD feature tracker.