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
- FLANN feature matching
- 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 of camera for it and JSON format configuration file.

Main success scenario:

1) User launches a program from the command line, specifying the path to a configuration file that contains the paths to video source, calibration matrix, output directory and other necessary parameters.

2) Program calls main video processing cycle with config parameters 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 and saves it as previous frame for further processing
3. Program fills a batch of frames. (Count of frames must be specified in config)
4. Program matches previously taken features with next frame's features and skips frames if there were not enough matched features count (Count of features must be specified in config).
   a. Program tries to find the best frame in the batch, judging by matched features count.
   b. If all frames are bad, we interrupt the cycle and restart it with a new initial global position.
5. Program estimates the motion and rotation between two matched frames.
6. Program constructs new global 3D points using triangulation and data estimated at previous two steps
7. Writes new camera position to **`<output_dir>/pose.txt`** and world 3D points to **`<output_dir>/points.txt`**
   a. The points in the files have format:

   ```
   X1 Y1 Z1
   X2 Y2 Z2
   ...
   ```

8. Update data for next iteration:

    a. Current frame extracted points vector is saved as a previous frame extracted points vector
    b. Skipped frames are deleted from the batch.
9. Visualize computed 3-dimensional points and camera positions.

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 "*invalid configuration*":

Trigger condition: configuration file specified by user doesn't contain some necessary field or contain field with incorrect type.

1) Execution interrupts and the program prints an error message about invalid/missed fields.

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 camera 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 configuration JSON where the calibration flag is set to `true`.

2) Program detects chessboard corners in the photos or video frames 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 with path "`calibrationPath`" from configuration JSON

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 "*invalid configuration*":

Trigger condition: configuration file specified by user doesn't contain some necessary field or contain field with incorrect type.

1) Execution interrupts and the program prints an error message about invalid/missed fields.

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 "Single visualization"

Actors: user.

Goals: to observe a 3D map with points and camera trajectory.

Precondition: user has .txt files with 3D points (`points.txt`), their colors (`colors.txt`), camera positions (`positions.txt`) and rotations (`rotations.txt`). They can be obtained by our algorithm or other source

Main success scenario:

1) The user runs the program from the command line with configuration JSON where the "`onlyViz`" flag is set to `true`.

2) Program reads files with data from directory specified in JSON config field "`outputDataDir`"

3) Program passes read data to visualizer entry point function

4) Program opens windows with 3D points. User can move in space using WASD, C for falling, Space for rising, and mouse with clicked LMB for rotation

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 "*invalid configuration*":

Trigger condition: configuration file specified by user doesn't contain some necessary field or contain field with incorrect type.

1) Execution interrupts and the program prints an error message about invalid/missed fields.

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

7. Non-functional requirements

7.1. Environment

The program is supported only on Linux with OpenCV 4.8 for C++ and additional modules described in deployment guide.

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 1 minute and resolution 1920 x 1080 was processed in 3 minutes on average (performance may vary greatly depending on the parameters).