SLAM: Simultaneous Localization and Mapping

Usama Ahmed Chatha
CSCI-4930-Autonomous Driving
Saint Louis University

I. INTRODUCTION

The report presents the SLMA system, a feature-based monocular simultaneous localization, and mapping. The SLAM system can perform in real-time, in small environments. It is a state-of-the-art feature-based SLAM approach. This SLAM system highly relies on the visual features observed from the environment. The features in the environment can be seen at different levels such as low-level features like points and edges, medium-level features like planes, and high-level features like labeled objects.

The implementation of the SLAM system depends on the sensors, cameras, and the data accuracy of the data collection. We used a feature-based approach, where conventional vision sensors such as monocular, depth, or stereo cameras are used to capture the environment. The proposed system captured the environment image by using the camera which is able to record a high-resolution image with rich details. The system extracts the features and key points from each frame. Based on the features, the best frame of the environment will be selected, and then map the feature between the keyframes. By using the best features and keyframes, the SLAM system will build a local map of the environment. The system used performs the localization to know the current position where the AV itself is. In the end, the system will build the driving trajectory of the autonomous vehicle.

II. System Design

A. Design Overview

1- Convert to Grayscale: First, for the SLAM system, we need to convert the BGR (Blue, Green, Red) frames that are captured by the camera of the surrounding environment into a grayscale image. we are using the cvtColor() function from the cv2 library that takes the colored image and converts it to the grayscale and returns the image.

2- Features Extraction: The system extracts the features from each. The features can be the points, edges, planes, and label objects. Our proposed system used the fast feature extraction method and ORB feature extractions to get the features from the grayscale image. This system uses the Brute force algorithm to optimize the keyframes. The threshold is set for each frame. It will help to select the keyframes from a set of frames and help the SLAM system to get an Optimized result.

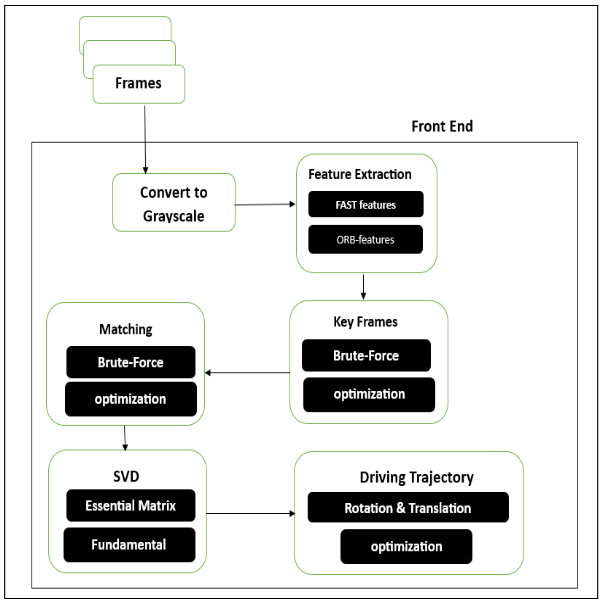


Fig. 1. System Architecture .

3- Matching: In the process of matching, the key points of each frame are matched with the key points of the other frame. Based on these matched points the SLAM will develop a local map of the surrounding environment. The Brute force algorithm is used to optimized the matches between the different frames to find the best keyframe sequence for the better result of SLAM system.

4- SVD: After matching and selecting the key frames the SLAM system calculates the essential matrix and the fundamental matrix. By estimating the essential and fundamental matrices, SLAM algorithms can track the robot's movement and map the environment accurately, even in the presence of noise and uncertainty.

5- Driving Trajectory: At last, the system will calculate the rotation and translation matrix. Rotation matrices are used to represent the orientation of the AV, while translation matrices are used to represent its position. When the robot moves, these matrices are updated to reflect its new position and orientation relative to its starting point. Based on the rotation and the translation Matrix, the driving trajectory for the autonomous vehicle will be predicted.

III. System work Flow

We are using the feature-based SLAM system approach. The first step in SLAM is to acquire data from the surrounding environments in the form of BGR images by using cameras. The system will convert the colored image to a grayscale image. Once the image is converted to grayscale, the system will extract the feature points. First, the fast feature extractor will extract the feature points from the grayscale image. Then the ORB feature extractor will extract the features from it. Based on the extracted feature and the number of features in each frame, the system will select the optimized frames with the useful number of features that will help the system to get a good matching and mapping of the surrounding. The brute force algorithm will optimize the feature that will help the SLAM to get the accurate results. In next step, the SLAM algorithm matches the extracted features from different sensor readings to create a consistent map. This is done by using techniques of feature matching or scan matching. After that the system will calculate the fundamental and the essential matrix. The system will get the rotation and translation matrix and based on this it will build a local map of the surrounding environment and predict the driving trajectory.

IV. Detailed system design

A. *Grayscale image*

In the SLAM system, it is essential and useful to convert multi-color frames into grayscale images. To convert the BGR frames, our system is using the cvtColor function from cv2 library. It takes the color image and converts it to the grayscale image. The are some important factors for which we need to convert each frame into a grayscale image. Firstly, grayscale images contain only one channel of intensity values, which can simplify the feature extraction process. Many feature extraction algorithms, such as the popular SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features) algorithms, are designed to operate on grayscale images. By converting a BGR frame to grayscale, we can reduce the computational complexity of the feature extraction algorithm and potentially increase its accuracy and robustness. Secondly, grayscale images can be more robust to changes in illumination or color variations. In color images, changes in lighting conditions or color can cause variations in pixel values across the different color channels, which can make feature extraction more difficult. Grayscale images, on the other hand, are more invariant to such changes, as they only contain intensity values. This can improve the reliability and consistency of the feature extraction algorithm.

B. *Fast Features Extraction*

The FAST algorithm is a popular feature detection method that finds critical areas in an image. For tasks like picture matching, object recognition, and tracking, key points are discrete visual properties that are independent of image scale, rotation, and translation. The function creates a FastFeatureDetector object with a number of parameters that can be adjusted to control how the feature detector behaves. These parameters include the feature detection threshold, the neighborhood type, and whether non-maximum suppression is used to remove redundant features. The detect() method, which accepts an image as input and returns a list of Key Point objects representing the detected key points, can be used to use the FastFeatureDetector object to find key points in an image after it has been produced. The length of key features in each frame is mentioned as a threshold to get the keyframe that provides better results. In this system, the number of features is set as 7000.

C. *ORB Features Extraction*

In SLAM, ORB feature extraction is utilized to find details in picture or video frames that can be used to gauge camera or robot motion and create an environment map. The FAST Features from Accelerated Segment Test corner detection algorithm and the BRIEF Binary Robust Independent Elementary Features descriptor algorithm are combined in the ORB feature extraction algorithm. The BRIEF method is used to describe the local picture patches around each key point as a binary code, and the FAST algorithm is used to identify key points in the image. The binary codes are made to be resistant to variations in lighting, noise, and other image modifications. The orb object used in the code snippet is an instance of the cv2 ORB create() class, which is an implementation of the ORB feature extraction algorithm provided by the OpenCV library. The cv2 drawKeypoints() function is used to visualize the key points detected by the ORB algorithm on the input image.

D. *Matching*

At this point, the feature points of the frames will be matched to get the keyframes. The keyframes selection is the critical point of the SLMA system. The selection of good keyframes will give a good driving trajectory. Our SLAM system is using the brute force algorithm for the optimization of the keyframes. The Brute-Force Matcher (BFMatcher) is frequently used in SLAM (Simultaneous Localization and Mapping) to match the descriptors of features in successive camera frames. The matching procedure is used to triangulate the 3D positions of the observed features and to estimate the camera motion between the two frames. For feature matching in SLAM, the BFMatcher is a straightforward and efficient algorithm, especially for small-scale and real-time applications. It evaluates the matches using a distance metric like Euclidean distance or Hamming distance, comparing each descriptor from the first frame with every descriptor in the second frame and returning the best match.

E. *Essential and Fundamental matrix*

To calculate the fundamental matrix and Essential matrix the system is using a function cv findFundamentalMat with parameter cv FM RANSAC and cv findEssentialMat with parameter cv.FM RANSAC which are the OpenCV library that computes the fundamental matrix and Essential matrix between two sets of corresponding points in two different images. The fundamental matrix and essential matrix are estimated using the RANSAC technique by iteratively choosing random subsets of the matching points in the two images and fitting a fundamental matrix to them. The best estimate of the fundamental matrix is chosen from the subset with the greatest number of inliers, or points that meet the above equation. The notions of essential and fundamental matrices are crucial to SLAM since they aid in determining the relative pose between two perspectives of a scene. In contrast to the fundamental matrix, which relates the picture coordinates of corresponding spots in the two views, the essential matrix connects two camera perspectives of the same scene. For the purpose of creating an environment map in SLAM, the relative posture between two views of a scene is estimated using the crucial matrix. As opposed to this, the fundamental matrix is utilized to ascertain the correspondences between the features in various viewpoints, which is crucial for establishing the robot's position and orientation inside the environment.

F. Rotation and Translation Matrix

At the last step, the SLAM system will calculate the rotation and translation matrix. it takes the current position of the camera and predicts the driving trajectory. Rotation and Translation matrices are essential for tracking the robot's position and orientation over time as it moves through its environment and builds a map of its surroundings. In particular, translation matrices are used to represent the robot's position and rotation matrices are used to indicate the robot's orientation in space. These matrices are updated to reflect the robot's new position and orientation in relation to its starting point when it moves. In order to precisely track the robot's movement and create a map of its environment, SLAM uses rotation and translation matrices to combine input from a range of sensors, including lidar, cameras, and inertial measurement units (IMUs). These matrices also give the system the ability to take into account any inaccuracies or uncertainties in the sensor data, resulting in more precise mapping. Based on the current camera position, pervious point rotation, and translation matrix, the SLAM will predict the next point of AV and build the driving trajectory.

V. Experiment Results

A. Kitti Dataset

For the experiment, we used the Kitti dataset as sohwn in figure 2.

B. Convert to Gray-scale

The BGR image converted to gray scale image as shown in figure 3.

C. Fast Feature Extraction

The fast feature extractor extracts the features from gray scale image as shown in figure 4

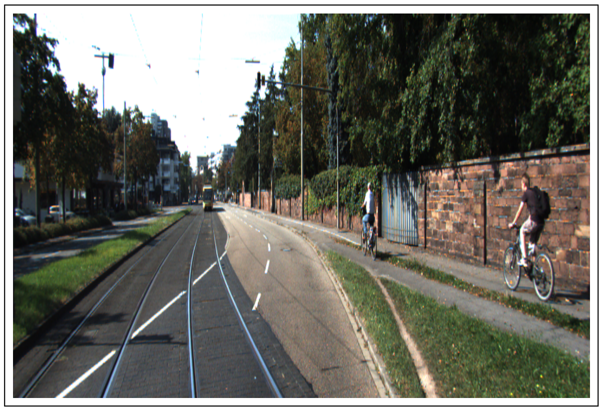


Fig. 2. kitti Dataset Frame.

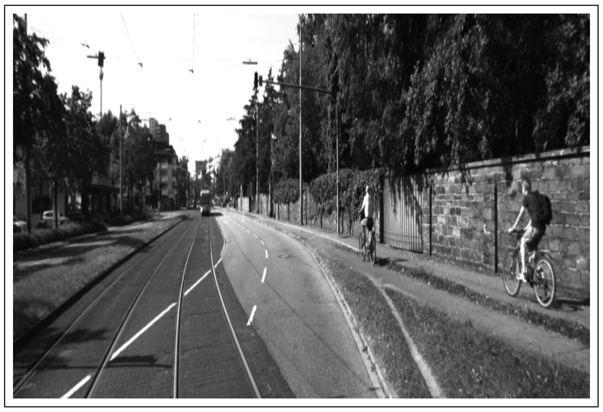


Fig. 3. Gray-scale image.

D. ORB Feature Extraction

The ORB feature extractor extracts the features from gray scale image as shown in figure 5

E. keypoints matching

The key points match with between the different frames as shown in figure 6.

F. Matrices

The essential matrix, fundameantal matrix, Rotation matrix, and translation matrix of the frames as shown in figure 7.

subsectionDriving Trajectory

The driving trajectory will be better by smartly selecting the key frames. The current driving trajectory is shown in figure 8.

VI. Conclusions

The SLAM system with feature-based approach highly depends on the key points in each frame and the patterns of different frames. The accuracy in the arrangement and the selection of key frames make the driving trajectory better and make the autonomous vehicle smarter.

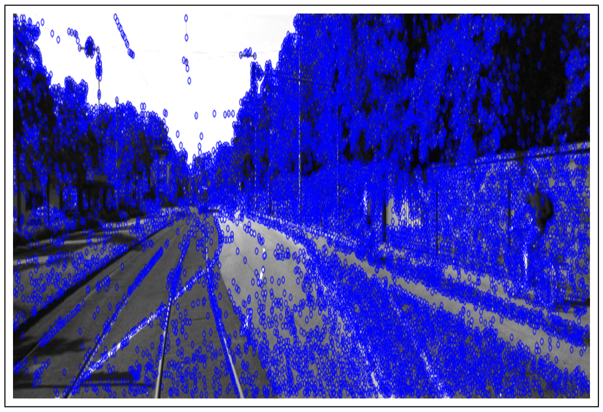
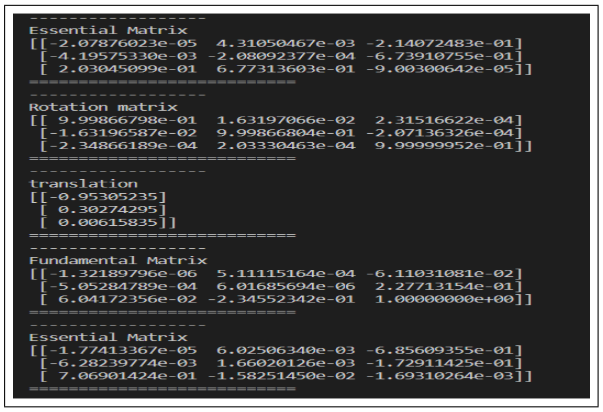


Fig. 4. Fast feature Extraction.



```
------------------
Essential Matrix
[[-2.07876023e-05  4.31050467e-03 -2.14072483e-01]
 [-4.19575330e-03 -2.08092377e-04 -6.73910755e-01]
 [ 2.03045099e-01  6.77313603e-01 -9.00300642e-05]]
===========================
------------------
Rotation matrix
[[ 9.99866798e-01  1.63197066e-02  2.31516622e-04]
 [-1.63196587e-02  9.99866804e-01 -2.07136326e-04]
 [-2.34866189e-04  2.03330463e-04  9.99999952e-01]]
===========================
------------------
translation
[[-0.95305235]
 [ 0.30274295]
 [ 0.00615835]]
===========================
------------------
Fundamental Matrix
[[-1.32189796e-06  5.11115164e-04 -6.11031081e-02]
 [-5.05284789e-04  6.01685694e-06  2.27713154e-01]
 [ 6.04172356e-02 -2.34552342e-01  1.00000000e+00]]
===========================
------------------
Essential Matrix
[[-1.77413367e-05  6.02506340e-03 -6.85609355e-01]
 [-6.28239774e-03  1.66020126e-03 -1.72911425e-01]
 [ 7.06901424e-01 -1.58251450e-02 -1.69310264e-03]]
===========================
```

Fig. 7. E, F, R, T of frames.

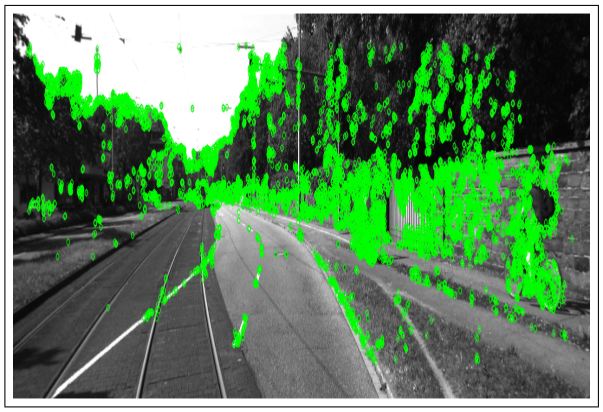


Fig. 5. ORB feature Extraction.

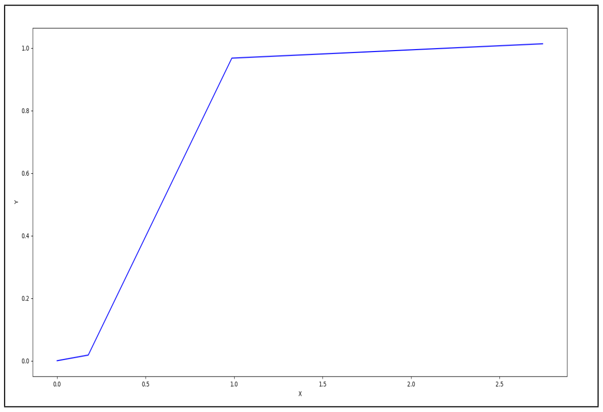


Fig. 8. Driving Trajectory.

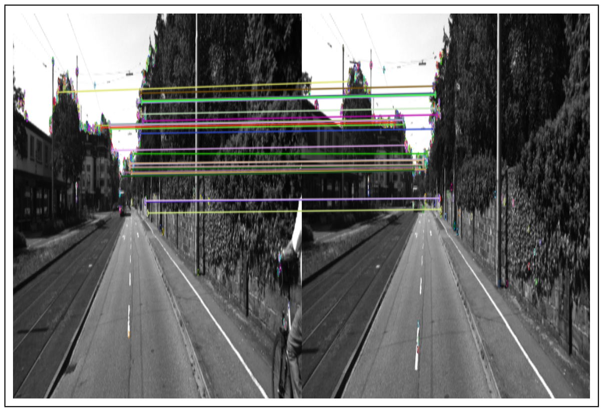


Fig. 6. Key points matching between frames.