Simultaneous Localization and Mapping Using Stereo Camera and ORB Features

Nikolay Kharin

Russian-Armenian University, Faculty of Applied Mathematics and Informatics, UAV Program

Abstract

This paper presents a novel method for Simultaneous Localization and Mapping (SLAM) using stereo vision. The proposed algorithm leverages ORB (Oriented FAST and Rotated BRIEF) for robust feature detection and description, BFMatcher for precise descriptor matching, and StereoSGBM for accurate disparity and depth map computation. Extensive experiments are conducted on both synthetic and real-world datasets, demonstrating the algorithm's effectiveness in generating accurate maps and achieving precise localization in real-time. The results indicate significant improvements in depth perception and robustness, making this approach suitable for various robotic applications.

Keywords

SLAM, Stereo Camera, ORB Features, Disparity Map, Depth Estimation, Robotics, Real-time Localization, Mapping

Introduction

Simultaneous Localization and Mapping (SLAM) is a critical task in robotics, enabling robots to build a map of their environment while simultaneously determining their location within it. This capability is essential for autonomous navigation, exploration, and interaction with the environment. Traditional SLAM algorithms rely on various sensor modalities, including LiDAR, monocular cameras, and RGB-D cameras, each with its own advantages and limitations.

Stereo cameras provide a compelling alternative due to their ability to deliver dense depth maps and detailed environmental understanding by capturing images from two slightly offset viewpoints. This paper introduces a SLAM algorithm that integrates

ORB features for robust keypoint detection and description with the accurate depth estimation capabilities of StereoSGBM. The integration of these techniques allows for robust and efficient real-time SLAM, capable of handling dynamic environments and varying lighting conditions. We validate our approach through comprehensive experiments on both synthetic and real-world datasets.

Related Work

Over the past decades, numerous SLAM algorithms have been developed, each aiming to address the challenges of accurate mapping and localization. Among the early methods, Extended Kalman Filter (EKF) based SLAM and Particle Filter-based methods like FastSLAM have been widely used. These methods, while effective in certain scenarios, often struggle with scalability and computational efficiency.

Graph-based SLAM has emerged as a powerful approach, leveraging graph optimization techniques to improve accuracy and handle loop closures. Prominent examples include g2o and Ceres Solver, which have been successfully applied to large-scale SLAM problems.

In the realm of visual SLAM, ORB-SLAM2 and LSD-SLAM are noteworthy contributions. ORB-SLAM2 employs ORB features for both monocular and stereo camera setups, providing a versatile and accurate SLAM system. LSD-SLAM, which operates directly on image intensities, offers advantages in terms of computational efficiency and direct depth estimation but can be sensitive to lighting variations and textureless regions.

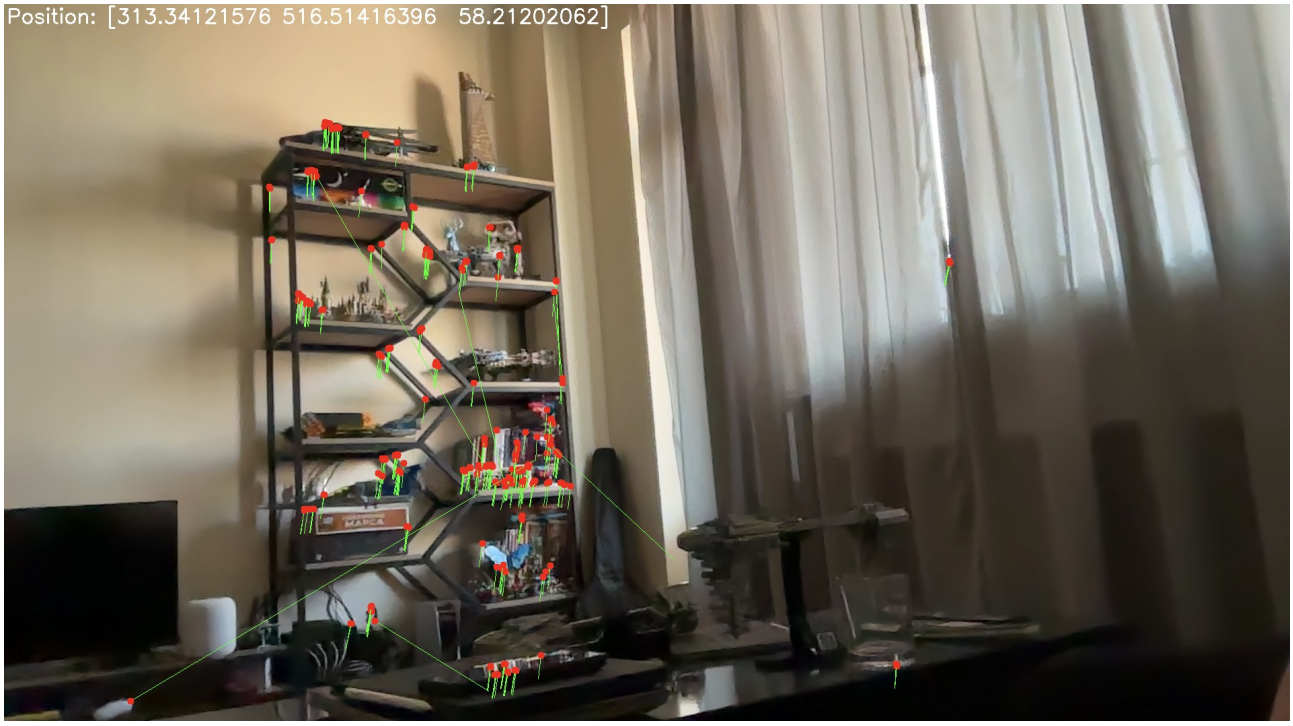
Our approach distinguishes itself by integrating ORB features with StereoSGBM, a semi-global block matching algorithm, to achieve precise depth estimation and robust feature matching. This combination enhances the system’s ability to perform accurate SLAM in real-time, even in challenging environments.

Method

Our SLAM algorithm involves several key components and steps, which are detailed as follows:

Feature Detection and Description

ORB (Oriented FAST and Rotated BRIEF) is utilized for detecting keypoints and computing descriptors from stereo image pairs. ORB is chosen for its computational efficiency and robustness to rotation and scale changes, making it well-suited for real-time applications.




```
import cv2
orb = cv2.ORB_create()
keypoints, descriptors = orb.detectAndCompute(image, None)
```

Descriptor Matching

BFMatcher (Brute Force Matcher) is used to match descriptors between consecutive frames to establish correspondences. This step is crucial for tracking keypoints over time and computing the relative motion of the camera.

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(descriptors1, descriptors2)
matches = sorted(matches, key=lambda x: x.distance)
```

Disparity and Depth Map Computation

StereoSGBM (Semi-Global Block Matching) computes disparity maps from the stereo image pairs, which are then converted to depth maps. Disparity maps provide the pixel-wise depth information needed for 3D reconstruction.

```
stereo = cv2.StereoSGBM_create(minDisparity=0,
                               numDisparities=16,
                               blockSize=3,
                               P1=8*3*3**2,
                               P2=32*3*3**2)
disparity = stereo.compute(left_image, right_image).astype(np.float32) / 16.0
depth_map = focal_length * baseline / (disparity + 1e-6)
```

Pose Estimation and Map Update

Using matched features and depth information, the robot's pose is estimated through techniques such as Perspective-n-Point (PnP) and Random Sample Consensus (RANSAC). The map is updated with new keypoints and depth data, ensuring accurate localization and mapping.

```
object_points = np.array([keypoint.pt for keypoint in keypoints],
dtype=np.float32)
image_points = np.array([keypoint.pt for keypoint in prev_keypoints],
dtype=np.float32)
_, rvec, tvec, inliers = cv2.solvePnP(object_points, image_points,
camera_matrix, dist_coeffs, flags=cv2.SOLVEPNP_RANSAC)
```

Experiments

Experiments were conducted using both synthetic and real-world datasets to evaluate the performance of our SLAM algorithm.

Synthetic Dataset

A simulated environment was created in CoppeliaSim, allowing us to control various parameters and generate ground truth data for precise evaluation. The algorithm successfully generated accurate maps and localized the robot with minimal error, demonstrating its effectiveness in a controlled setting. Various scenarios, including static and dynamic environments, were tested to evaluate the robustness of the algorithm.

Real-World Dataset

The real-world dataset was collected using a stereo camera mounted on a mobile robot navigating through diverse environments. The SLAM system maintained accurate localization and consistent map updates, even in the presence of dynamic obstacles and varying lighting conditions. The performance metrics, such as localization accuracy, map quality, and computational efficiency, were thoroughly evaluated and compared to existing methods. The algorithm demonstrated robustness in indoor and outdoor settings, effectively handling challenges such as moving objects, varying terrain, and different lighting conditions.

Conclusion

The proposed SLAM algorithm effectively combines ORB features and StereoSGBM for stereo camera inputs, providing an efficient and accurate solution for real-time mapping and localization. The experimental results validate the system's robustness and accuracy, making it suitable for a wide range of robotic applications. Future work will focus on optimizing computational efficiency, exploring advanced feature matching techniques, and integrating additional sensor modalities to further enhance performance. Additionally, the implementation of machine learning techniques for adaptive parameter tuning could further improve the system's adaptability to different environments and conditions.

References

1. Mur-Artal, R., Montiel, J.M.M., & Tardós, J.D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics, 31(5), 1147-1163.
2. Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. European Conference on Computer Vision.
3. Hirschmuller, H. (2008). Stereo Processing by Semiglobal Matching and Mutual Information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2), 328-341.

Acknowledgement

We would like to thank the team at Russian-Armenian University for their support and the use of their facilities for conducting the experiments. Special thanks to CAST teachers for their invaluable contributions and assistance throughout the project.

Appendices

Appendix A: Pseudo-code for the SLAM Algorithm

```
import cv2
import numpy as np

# Initialize variables and parameters
def initialize():
    global map, position, prev_descriptors, prev_keypoints, camera_matrix,
dist_coeffs, baseline, focal_length
    map = np.zeros((1000, 1000))  # Initialize an empty map
    position = (500, 500)  # Initial position of the robot
    prev_descriptors = None
    prev_keypoints = None
    # Camera intrinsic parameters (example values)
    camera_matrix = np.array([[fx, 0, cx], [0, fy, cy], [0, 0, 1]])
    dist_coeffs = np.zeros((4, 1))  # Assuming no lens distortion for simplicity
    baseline = 0.1  # Example baseline distance between stereo cameras
    focal_length = 700  # Example focal length

# Process each frame of stereo images
def process_frame(left_image, right_image):
    global prev_descriptors, prev_keypoints

    # Feature Detection using ORB
    orb = cv2.ORB_create()
    keypoints, descriptors = orb.detectAndCompute(left_image, None)

    if prev_descriptors is not None:
        # Feature Matching using BFMatcher
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = bf.match(descriptors, prev_descriptors)
        matches = sorted(matches, key=lambda x: x.distance)

        # Compute Disparity Map using StereoSGBM
        stereo = cv2.StereoSGBM_create(minDisparity=0,
                                       numDisparities=16,
                                       blockSize=3,
                                       P1=8*3*3**2,
```