ORBSLAM3 on iPhone

M. Jia; C. Wang; G. Su; M. Li

Northeastern University

EECE5554 Final Project

Instructor: Dr. Hanumant Singh

Date: Apr. 27th, 2022

Abstract

To achieve accurate robotics perceptual capacity, researchers typically use multiple sensors to obtain comprehensive and precise environmental information. These sensors are non-integrated, decentralized, and in most cases need to be configured separately. Researchers need to spend a great amount of time on the installation and configuration of these sensors. Meanwhile, the iPhone is a highly integrated electronic device with multiple precise sensors, but few researchers have used the iPhone as a hardware device for algorithm development because of the closed source properties of IOS. To this end, we aim to develop an IOS-based ORBSLAM algorithm that has demonstrated the exploitability of the iPhone as well as providing a new possibilities of research platform. On another point, The advanced perception sensor on the iphone gives some imagination to the future development of the IOS-based metaverse app. By using the iPhone, many interesting applications can be further extended, such as augmented reality, indoor localization, etc.

Keywords: Visual SLAM, Mobile device, Optimization

Catalog

Introduction

ORB-SLAM3 is a state-of-the-art visual SLAM. This feature-based SLAM creates a sparse mapping and gives accurate localization. Also, it contains modules such as place recognition, loop detection, bundle adjustment, etc. The process is divided into three phases: short-term phase, medium-phase, and long-term phase. However, it is hard to access ORBSLAM for users who are not familiar with Linux, computer vision, and Github. At the same time, the hardware like Realsense camera and IMU that ORBSLAM requires are hard to access. We create a streaming layer between iPhone and ORBSLAM on Ubuntu, so that everyone can play with the state-of-the-art SLAM via iPhone and a PC with pre-installed ORBSLAM.

Method in ORBSLAM3

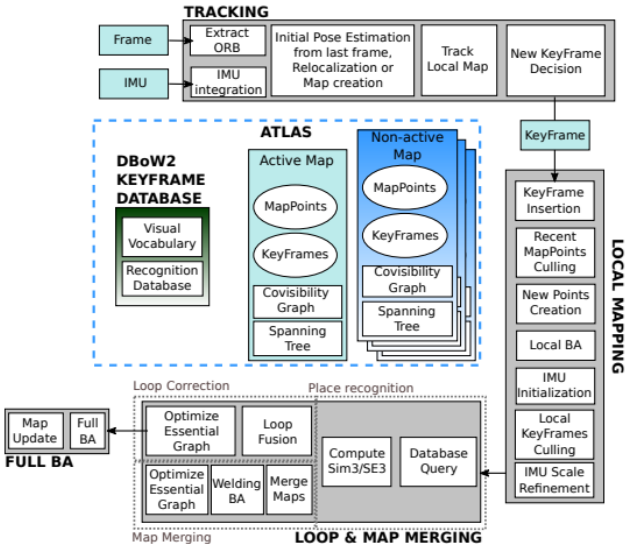

Figure 1. Main system components of ORB-SLAM3.

Short-term phase

For the short-term phase, the algorithm keeps tracking every frame to obtain relative transformation between the current frame and the last frame. By tracking ORB features in frames, we are able to calculate relative rotation and translation by P-n-P algorithm. If tracking is lost, the algorithm utilizes the place recognition module mentioned below to relocalize itself. A motion-only BA is also processed in this phase.

$$\{\mathbf{R}_{\mathrm{CW}}, {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}\} = \underset{\mathbf{R}_{\mathrm{CW}}, {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}}{\operatorname{argmin}} \sum_{j} \rho\left(\left\|\mathbf{x}_{\mathrm{C}}^{j} - \pi_{m}\left(\mathbf{R}_{\mathrm{CW}}\mathbf{X}_{\mathrm{W}}^{j} + {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}\right)\right\|_{\Sigma_{\mathrm{j}}}^{2}\right)$$

Equation 1. Motion-only bundle adjustment of ORB-SLAM3.

Medium-term phase

ORBSLAM3 utilized local bundle adjustment to minimize reprojection error to get more accurate pose and mapping points. The error term is listed in Equation 1. Bundle adjustment is a computationally heavy process to run, so it is only used in the medium-term phase as a special case of the generic BA[1]. In this phase, only keyframes are optimized due to consideration of memory and computation cost. Keyframes are selected based on several criteria[2]:

1. More than 20 frames must have passed from the last global relocalization.

2. Local mapping is idle, or more than 20 frames have passed from the last keyframe insertion.

3. Current frame tracks at least 50 points.

4. Current frame tracks less than 90% points than Kref.

$$\{\mathbf{X}_{\mathrm{W}}^{p}, \mathbf{R}_{\mathrm{CW}}^{l}, {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}^{l} | p \in \mathcal{P}_{L}, l \in \mathcal{K}_{L}\} = \underset{\mathbf{X}_{\mathrm{W}}^{p}, \mathbf{R}_{\mathrm{CW}}^{l}, {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}^{l}}{\operatorname{argmin}} \sum_{k \in \mathcal{K}_{L} \cup \mathcal{K}_{F}} \sum_{j \in \mathcal{P}_{k}} \rho\left(\left\|\mathbf{x}_{k}^{j} - \pi_{m}\left(\mathbf{R}_{\mathrm{CW}}^{k}\mathbf{X}_{\mathrm{W}}^{j} + {}_{\mathrm{C}}\mathbf{p}_{\mathrm{W}}^{k}\right)\right\|_{\Sigma_{k}^{j}}^{2}\right)$$

Equation 2. Optimization based on reprojection error

Long-term phase

In the long-term phase, two main components are presented, which are loop fusion and essential graph optimization. Different from ORBSLAM2, ORBSLAM3 added a full bundle adjustment. The error term and cost function for essential graph optimization are listed below. In order to construct a pose graph,

$$\mathbf{e}_{rel}(i, j) = \mathrm{Log}_{\mathrm{SE}(3)}\left(\hat{\mathbf{T}}_{\mathrm{CC}}^{ij}\mathbf{T}_{\mathrm{CW}}^{j}\mathbf{T}^{i^{-1}}{}_{\mathrm{CW}}\right)$$

Equation 3. Relative transformation error

$$C = \sum_{(i,j)\in X} \rho(||e_{rel}(i, j)||^{2}_{\Sigma ij})$$

Equation 4. Pose graph optimization

Place recognition module

In order to detect loops and do relocalization, the algorithm should have the ability to recognize places it visited before. The bag of words technique[3] is used to convert frames into feature vectors. To calculate the distance/similarity between frames, the algorithm uses L1 score listed below.

$$s(v_1, v_2) = 1 - \frac{1}{2}\left|\frac{v_1}{|v_1|} - \frac{v_2}{|v_2|}\right|$$

Equation 5. Similarity score between two bag of word vectors

Method in Streaming

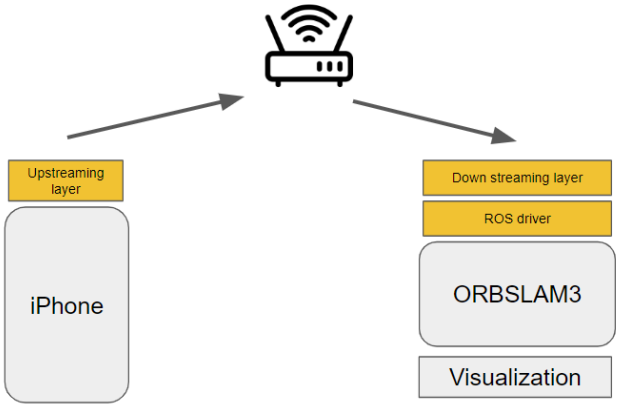

Figure 2. Streaming layers

Streaming layer

For the upstreaming layer, we use Droidcam to upload our video frames from our smartphones. A list of webcam applications is tested and we found Droidcam can be installed both on iPhone and Android. And, a ubuntu-based client is also supported by Droidcam, which is used as our down streaming layer. These two layers are connected by a WIFI router. For the Droidcam client on Ubuntu, several libraries are necessary, such as Gstreamer and Atlas.

ROS driver

Once we get real-time streaming from our phones, a ROS driver is needed for publishing our frames. We use video_stream_opencv to publish our video frame, by which we are able to define image size and frequency to keep it consistent with our down streaming layer.

On-device development

Dependencies modification

In order to fully use the functionality or ORB_SLAM, some dependencies need to be modified to better accommodate the Xcode environment. Those such as OpenCV, g2o, libjpeg can not be directly compiled on Xcode, so different methods have been used to address this problem. Some pre-compiled libraries were used such as g2o, libjpeg so that the project could call those libraries directly. And for the opencv framework, we replaced the original OpenCV folder by using the OpenCV for IOS to address errors appearing on compiling the self-contained library.

UI design

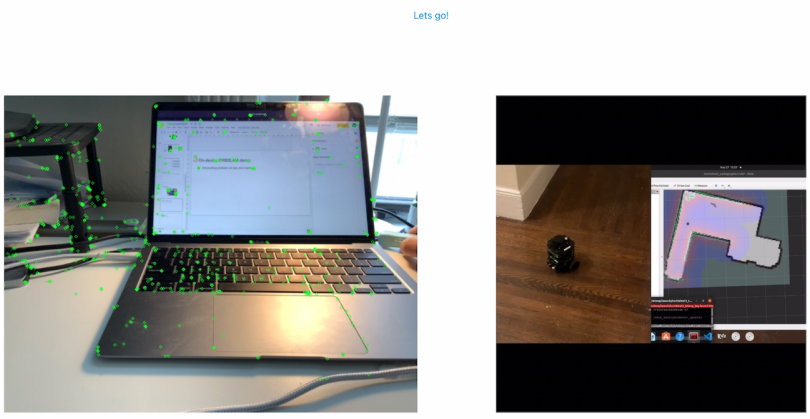

Figure 3. User Interface

In order to better demonstrate the results of the algorithm, we designed the UI according to the original author's design ideas, mainly divided the UI into two parts: the video stream play and the viewer which shows in the figure below. On the left side is the stream player which could play the camera video in real-time with a feature point attached. Due to the non-compilability of Pangolin on ISO, we have not achieved the functionality of the origin viewer yet. So the right side part of this app is reserved for the viewer which will be rewritten by using SwiftUI in the future.

Appendix

This appendix is mainly about the problems and solutions to code running. More problems with installation and library dependencies will show in the readME of this project.

A. Pangolin Visualization
"ORB_SLAM3::System SLAM(argv[1],argv[2],ORB_SLAM3::System::MONOCULAR, **true**)" the last parameter of the SLAM show the visualization status on(true) or off(false).
(Path: …/ORB_SLAM3/Examples/Monocular/mono_euroc.cc)

B. EuRoC Datasets
The **gravity acceleration** is assumed to be $9.81 m/s^2$ but Boston has a gravity of $9.80379 m/s^2$, which may affect the data of the z-axis.

C. Kitti Datasets
Dataset analysis: The ORBSLAM3 is usually tested on the EuRoc dataset, but our data is more compatible with the official dataset KITTI.
The algorithm will make the prediction of the routine with input video or ground of photos with a txt file with 8 parameters on each line, which is position and orientation: '**timestamp x y z q_x q_y q_z q_w**', for our evaluation, we use x y z for comparison.

D. ORB-SLAM3
Ran into error when building ORBSLAM3 using build.sh
Modify *build.sh* by: sed -i 's/++11/++14/g' CMakeLists.txt

E. XCode
- The iPhone 13 series does not work on both simulation and real machines.
- Need to set up the camera in private env parameters.
- 'Unable to install the app': go to the TARGETS -> select[your project name] -> General -> Frameworks,Libraries,and EmbeddedContent -> set the framework with [Do Not Embed]
- 'failed to prepare device for deployment' & 'This operation can fail if the version of the OS on the device is incompatible with the installed version of Xcode.': check XCode version, check iOS version, restart the XCode (computer) and iPhone
- Storyboard ID: https://www.raywenderlich.com/5055364-ios-storyboards-getting-started
- Xcode UIView.init(frame:) must be used from main thread only: Main Thread Checker https://medium.com/@trivediniki94/main-thread-checker-in-xcode-8b9f3f8ce10
- SWIFT_VERSION '3.0' is unsupported, supported versions are: 4.0, 4.2, 5.0. https://stackoverflow.com/questions/55366024/how-to-fix-swift-version-3-0-is-unsupported-supported-versions-are-4-0-4-2
- Link main storyboard with code: https://stackoverflow.com/questions/46553129/xcode-how-to-view-two-files-side-by-side
- Xcode has conflicting provisioning settings: https://stackoverflow.com/questions/42885122/xcode-has-conflicting-provisioning-settings
- Xcode error "Can't install application ... [appname].app requires the " z" capability which is not supported by [devicename] https://stackoverflow.com/questions/34272690/xcode-error-cant-install-application-appname-app-requires-the-z-capab

- Swift Initializer for conditional binding must have Optional type, not '[AVCaptureDevice]' https://stackoverflow.com/questions/47843403/swift-initializer-for-conditional-binding-must-have-optional-type-not-avcaptu

References

[1]Campos, Carlos, et al. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam." *IEEE Transactions on Robotics* 37.6 (2021): 1874-1890.

[2]Mur-Artal, Raúl, and Juan Domingo Tardós Solano. "Real-Time Accurate Visual SLAM with Place Recognition." *Ph. D Thesis* (2017).

[3]Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." *IEEE Transactions on Robotics* 28.5 (2012): 1188-1197.