NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF SIGNALS, CONTROL AND ROBOTICS
INTELLIGENT ROBOTICS AND AUTOMATION LABORATORY

SIMULTANEOUS LOCALIZATION AND MAPPING OF UNMANNED ROBOTIC VEHICLE WITH APPLICATIONS TO AUTONOMOUS NAVIGATION IN A CHALLENGING ENVIRONMENT

DIPLOMA THESIS

of

Ioannis Alamanos

Supervisor: Costas Tzafestas
Assoc. Prof. NTUA

Athens, October 2022

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF SIGNALS, CONTROL AND ROBOTICS
INTELLIGENT ROBOTICS AND AUTOMATION LABORATORY

SIMULTANEOUS LOCALIZATION AND MAPPING OF UNMANNED ROBOTIC VEHICLE WITH APPLICATIONS TO AUTONOMOUS NAVIGATION IN A CHALLENGING ENVIRONMENT

DIPLOMA THESIS

of

Ioannis Alamanos

Supervisor: Costas Tzafestas
Assoc. Prof. NTUA

Approved by the examining committee on 2022-10-31

.....	.....	.....
C. Tzafestas, Assoc. Prof. NTUA	P. Maragos, Prof. NTUA	H. Psilakis, Lect. NTUA

Athens, October 2022

.....
Ioannis Alamanos
Graduate Electrical and Computer Engineer

Abstract

In this thesis we propose ORB-LINE-SLAM, a real-time hybrid point-line and only-line based visual Simultaneous Localizing and Mapping (SLAM) system for stereo cameras which can work in standard CPUs. This work is based on the ORB-SLAM3[1] open-source library. The motivation of the present thesis was to infuse line segments into the core of ORB-SLAM3 so as to overcome the challenges that it faces in demanding environments in which point features are not sufficient to estimate precisely the pose of the camera. The first novelty of the system is the introduction of an experimentally tuned adapting factor which aims to achieve a more efficient fusion of point and line features, as compared to classical methods which weigh the participation of points and lines in the optimization process equally. This necessity emerged from the complicated nature of the optimizations used in the line SLAM which assume that the endpoints of corresponding line segments are stable, although they are actually drifted throughout the infinite line that the line segment defines. In this way, we minimize the accumulating error in frames plentiful in point correspondences while we still take advantage of the rich information that line segments provide in low-textured environments in which the system determines insufficient orb matches. Moreover, to the best of our knowledge, this is the first open-source[1] visual SLAM system that has the ability to effectively work by using line features exclusively. The biggest challenge related to only-line SLAM is that with the current existing line segment detectors and the restrictions that they set regarding the computational cost, the system can not define an adequate number of correspondences for the pose estimation. However, in our implementation by employing an efficient strategy for abstracting outliers and extracting lines from 2 scale levels with a sacrifice of computational cost, we significantly reinforce the efficacy of the system and partly overcome this impedance. Additional contributions of this thesis concern the expansion and adaptation of the components of ORB-SLAM3 to cope with line features, as well as the systematic comparison of different line feature detectors and error functions for the Bundle Adjustment (BA) process, to deduce which ones are the more effective for the visual SLAM problem. The results obtained from our experiments indicate that the proposed point-line method significantly improves the accuracy of ORB-SLAM3 in challenging conditions. Furthermore, the only-line method implemented in this work, despite its simplified nature, also manages to work in a standalone mode and to finish even difficult sequences with remarkable precision.

Keywords: SLAM, Bundle Adjustment, Localization, Mapping, Line Segment Features, Point Features, Loop Closing

[1]The source code in this work is publicly available at https://github.com/Giannis-Alamanos/ORB-LINE-SLAM

Περίληψη

Σε αυτή την διπλωματική εργασία παρουσιάζουμε το ORB-LINE-SLAM, ένα υβριδικό σύστημα ταυτόχρονης χαρτογράφησης και εκτίμησης θέσης (Simultaneous Localization and Mapping - SLAM) για stereo κάμερες, βασισμένο σε σημεία και γραμμές ή αποκλειστικά σε γραμμές, το όποιο έχει τη δυνατότητα να λειτουργήσει σε πραγματικό χρόνο σε συμβατικές CPUs. Η δουλειά αυτή είναι βασισμένη στην ORB-SLAM3 [1] open-source βιβλιοθήκη. Το κίνητρο της παρούσας διπλωματικής ήταν να ενσωματώσουμε τα ευθύγραμμα τμήματα στον πυρήνα του ORB-SLAM3 ώστε να δώσουμε λύση στις προκλήσεις που το σύστημα αντιμετωπίζει σε απαιτητικά περιβάλλοντα όπου τα σημεία δεν επαρκούν για να υπολογιστεί με ακρίβεια η πόζα της κάμερας. Η πρώτη καινοτομία του συστήματος είναι η εισαγωγή ενός πειραματικά ρυθμισμένου προσαρμοστικού συντελεστή ο οποίος στοχεύει στο να πετύχει μια πιο αποδοτική συγχώνευση σημείων και γραμμών σε σύγκριση με άλλες κλασικές μεθόδους που τα σταθμίζουν το ίδιο στην διαδικασία βελτιστοποίησης. Αυτή η αναγκαιότητα προέκυψε από την ιδιόμορφη φύση των βελτιστοποιήσεων που χρησιμοποιούνται στο SLAM με γραμμές, οι οποίες υποθέτουν ότι τα ακραία σημεία των ταιριασμένων ευθύγραμμων τμημάτων είναι σταθερά, ενώ στην πραγματικότητα είναι μετατοπισμένα στην ευθεία που ορίζει ένα από τα δύο ευθύγραμμα τμήματα. Με αυτό τον τρόπο, ελαχιστοποιούμε το συσσωρευτικό σφάλμα στις εικόνες που υπάρχει αφθονία σε αντιστοιχίες σημείων ενώ διατηρούμε τη δυνατότητα εκμετάλλευσης της πλούσιας πληροφορίας που τα ευθύγραμμα τμήματα παρέχουν σε χαμηλής υφής περιβάλλοντα στα οποία το σύστημα προσδιορίζει ανεπαρκή αριθμό ORB αντιστοιχίσεων. Ακόμη, από όσο γνωρίζουμε, αυτό είναι το πρώτο open-source[2] σύστημα οπτικού SLAM που έχει την ικανότητα να λειτουργήσει αποδοτικά αποκλειστικά με τη χρήση γραμμών. Επιπρόσθετες συνεισφορές αυτής της διπλωματικής αφορούν την επέκταση και την προσαρμογή των δομικών στοιχείων του ORB-SLAM3 όσον αφορά τις γραμμές, όπως επίσης και τη συστηματική σύγκριση διαφορετικών αλγορίθμων για εντοπισμό γραμμών αλλά και συναρτήσεων σφάλματος για τη διαδικασία του Bundle Adjustment (BA), ώστε να καταλήξουμε ποιες είναι οι πιο αποτελεσματικές για το πρόβλημα του οπτικού SLAM. Τα αποτελέσματα από τα πειράματά μας υποδεικνύουν ότι η προτεινόμενη μέθοδος που χρησιμοποιεί σημεία και γραμμές βελτιώνει σημαντικά την ακρίβεια του ORB-SLAM3 σε απαιτητικές συνθήκες. Επιπλέον, η μέθοδος μόνο με γραμμές που αναπτύχθηκε σε αυτή την εργασία, ανεξάρτητα από την απλοποιημένη φύση της, καταφέρνει σε αυτόνομη λειτουργία να ολοκληρώσει ακόμη και δύσκολες ακολουθίες εικόνων με αξιοσημείωτη ακρίβεια.

Keywords: SLAM, Bundle Adjustment, Εκτίμηση Θέσης, Χαρτογράφηση, Χαρακτηριστικά ως Ευθύγραμμα Τμήματα, Σημειακά Χαρακτηριστικά, Κλείσιμο Βρόχου

[2]Ο σχετικός κώδικας με την εργασία είναι διαθέσιμος στο https://github.com/Giannis-Alamanos/ORB-LINE-SLAM

Acknowledgments

First and foremost, I would like to thank my supervisor Assoc. Prof. C. Tzafestas for inspiring and motivating me as an undergraduate student through instructive lectures and fruitful conversations. Besides, his guidance, support and meticulous reviews of the work played a catalytic role in the formation of the thesis. In addition, I am grateful to Georgios Thanellas for his innovative ideas and our constructive conversations which encouraged and guided me from the beginning of the thesis.

Finally, I would like to thank my friends for being a source of inspiration for me and making these past few years unforgettable, and especially my family, to whom I dedicate this work to, who have always been by my side by unconditionally encouraging, supporting, and believing in me.

Contents

List of Figures

List of Tables

List of Algorithms

Εκτεταμένη Περίληψη

1. Εισαγωγή

Η σύγχρονη ταυτόχρονη χαρτογράφηση και εκτίμηση θέσης (Simultaneous Localization and Mapping - SLAM), περιλαμβάνει την εκτίμηση της πόζας ενός ρομποτικού πράκτορα σε συνδυασμό με τη δημιουργία ενός χάρτη χαρακτηριστικών του περιβάλλοντος χώρου. Η δημοφιλία που έχει αποκτήσει το αντικείμενο αυτό τις τελευταίες δεκαετίες είναι αναμενόμενη αν κάποιος σκεφτεί το ευρύ φάσμα πεδίων που συνδυάζει αλλά και εφαρμογών στις οποίες μπορεί να δώσει λύση. Στο χαμηλότερο επίπεδο, το SLAM συσχετίζεται με επιστημονικά πεδία όπως η όραση υπολογιστών, η επεξεργασία σήματος, και η τεχνητή νοημοσύνη, ενώ σε υψηλότερο επίπεδο συνδυάζει ανάμεικτους τομείς όπως για παράδειγμα γεωμετρία, θεωρία δικτύων, βελτιστοποίηση, και πιθανότητες. Επιπροσθέτως, είναι συνυφασμένο και με πρακτικά θέματα όπως η βαθμονόμηση (calibration) των αισθητήρων και η δημιουργία και διασύνδεση του συστήματος. Οι εφαρμογές με τις οποίες συνδέεται το SLAM ποικίλλουν και έγκεινται σε διάφορες προκλήσεις της σύγχρονης ζωής. Συγκεκριμένα, είναι εξαιρετικά χρήσιμο για εφαρμογές τύπου έρευνα-και-διάσωση (search-and-rescue) όπως η εξερεύνηση περιοχών που δεν είναι προσπελάσιμες από τον άνθρωπο, για παράδειγμα εξερεύνηση του διαστήματος ή επιθεώρηση λατομείων και ετοιμόρροπων ορυχείων ή ακόμη και διάσωση ανθρώπων σε δύσβατες περιοχές, για την μεταφορά ατόμων με ειδικές ανάγκες μέσω αυτόνομων οχημάτων, και για εφαρμογές ρομποτικής επιθεώρησης υποδομών (robotic inspection) όπως η χαμηλού κόστους επιθεώρηση για συντήρηση, παραδείγματος χάρη η εξέταση ζημιών σε βιομηχανικές περιοχές ή διάβρωσης σε αμπάρια πλοίων. Αδιαμφισβήτητα, όλα τα παραπάνω στοιχεία που σχετίζεται το SLAM, το καθιστούν μία από τις μεγαλύτερες προκλήσεις στο πεδίο της σύγχρονης ρομποτικής.

Όπως είναι λογικό, οι πρώτες προσεγγίσεις σχετικά με το πεδίο του SLAM, αφορούσαν κυρίως στατικά περιβάλλονται και εστίαζαν σε πιθανοτικές μεθόδους για τον υπολογισμό της πόζας του ρομποτικού πράκτορα. Στη συνέχεια, η συστηματική έρευνα σε συνδυασμό με τη διαθεσιμότητα μεγάλων χώρων μνήμης αποθήκευσης, κατέστησαν δυνατή την αποθήκευση μεγάλου αριθμού παρατηρήσεων του ρομποτικού πράκτορα αλλά και την υιοθέτηση κοστοβόρων μη γραμμικών αλγορίθμων βελτιστοποίησης που μπόρεσαν να δώσουν ακριβή αποτελέσματα για την εκτίμηση της πόζας και τη χαρτογράφηση του χάρτη. Οι πιο εξελιγμένοι SLAM αλγόριθμοι μπορούν να διαχωριστούν σε δύο κατηγορίες ανάλογα με τον κύριο αισθητήρα που χρησιμοποιούν, και συγκεκριμένα αυτές είναι: το LiDAR SLAM, και το οπτικό SLAM. Το LiDAR SLAM βασίζεται στη χρήση ενός LiDAR και αποτελεί την πιο αποδοτική επιλογή στον τομέα καθώς εμφανίζει ανεκτικότητα στην έλλειψη ή την εναλλαγή του φωτισμού αλλά και στην υφή και μορφή του περιβάλλοντος χώρου. Από την άλλη μεριά, το οπτικό SLAM αποτελεί την πλέον ενδεδειγμένη επιλογή στον κλάδο από άποψη

σχέσης ποιότητας και τιμής. Πιο συγκεκριμένα, εκτός του γεγονότος ότι μέχρι και οι πιο εξελιγμένες κάμερες είναι αρκετά φθηνότερες από ένα αξιόπιστο LiDAR, η κάμερα απαιτεί σημαντικά λιγότερη ισχύ ενώ ταυτόχρονα παρέχει ακριβή αποτελέσματα. Για τον λόγο αυτό, το οπτικό SLAM είναι κατάλληλο για πολύ περισσότερες πρακτικές εφαρμογές και αποτελεί τη συνήθη επιλογή στο πεδίο αυτό. Το οπτικό SLAM χωρίζεται σε δύο κατηγορίες ανάλογα με τον τύπο του σφάλματος που ελαχιστοποιεί έτσι ώστε να εκτιμήσει την πόζα και να βελτιστοποιήσει το χάρτη. Αυτές είναι: οι άμεσες μέθοδοι (direct methods), όπου το σύστημα βελτιστοποιεί ελαχιστοποιώντας το φωτομετρικό σφάλμα, και οι μέθοδοι βασισμένες σε χαρακτηριστικά της εικόνας (feature-based methods), στις οποίες ο αλγόριθμος βελτιστοποιεί ελαχιστοποιώντας το σφάλμα μεταξύ ζευγοποιημένων χαρακτηριστικών. Τα συνήθη χαρακτηριστικά που χρησιμοποιούνται για το SLAM είναι τα σημεία. Παρόλα αυτά, τα τελευταία χρόνια έχουν αρχίσει και εισάγονται και άλλα γεωμετρικά χαρακτηριστικά, όπως κάναμε και εμείς στην υλοποίησή μας με τα ευθύγραμμα τμήματα, ώστε να προσδώσουν μεγαλύτερη ανθεκτικότητα στο σύστημα σε απαιτητικές συνθήκες. Μέσω αυτής της μεθόδου αλλά και χρησιμοποιώντας και άλλους αισθητήρες, όπως για παράδειγμα Inertial Measurement Units (IMU), το SLAM τα τελευταία χρόνια έχει εξελιχθεί αρκετά και έχουν αρχίσει και δίνονται λύσεις σε κάποιες από τις προκλήσεις που αναφέραμε προηγουμένως.

Στην παρούσα διπλωματική, ενσωματώνουμε τα ευθύγραμμα τμήματα ως χαρακτηριστικά της εικόνας στο stereo σύστημα του ORB-SLAM3[1] και παρουσιάζουμε το ORB-LINE-SLAM. Ο κύριος στόχος του project ήταν να σχεδιάσουμε ένα βελτιωμένο υβριδικό σύστημα στο οποίο τα ευθύγραμμα τμήματα θα έχουν ένα βοηθητικό ρόλο ως προς τα σημειακά χαρακτηριστικά έτσι ώστε να ανυψωθεί η ανθεκτικότητα και αποδοτικότητα του συστήματος σε απαιτητικές συνθήκες. Η κύρια συνεισφορά αυτής της εργασίας μπορεί να αποτυπωθεί επιγραμματικά ως εξής:

- Μια καινοτόμα στρατηγική σταθμισμένης ενοποίησης των σημείων και των γραμμών μέσω ενός πειραματικά ρυθμισμένου προσαρμοστικού παράγοντα.
- Το πρώτο open-source SLAM σύστημα το οποίο είναι ικανό να λειτουργήσει αποκλειστικά με τη χρήση ευθύγραμμων τμημάτων ως χαρακτηριστικά της εικόνας.
- Η ενσωμάτωση των ευθύγραμμων τμημάτων στον stereo αλγόριθμο του ORB-SLAM3.
- Ένα open-source σύστημα σημείων και γραμμών που ξεπερνάει σε απόδοση τη stereo εκδοχή του ORB-SLAM3, η οποία είναι επί του παρόντος η βέλτιστη stereo υλοποίηση διαθέσιμη open-source.

2. Σχετική Έρευνα

Το πρόβλημα του προσδιορισμού της πόζας ενός ρομποτικού πράκτορα αλλά και της χαρτογράφησης του περιβάλλοντα χώρου δεν είναι νέο στην επιστημονική κοινότητα. Ανά τα χρόνια, έχουν γίνει αρκετές προσπάθειες για την δημιουργία αποδοτικών αλγορίθμων, εκ των οποίων οι προγενέστεροι εστίασαν αποκλειστικά στον υπολογισμό της πόζας, ενώ οι μεταγενέστεροι προσέγγισαν συνολικά το πρόβλημα του SLAM. Ξεκινώντας από πιθανοτικές μεθόδους και καταλήγοντας σε σύγχρονα συστήματα, θα εμβαθύνουμε σε μερικούς από τους βασικότερους αλγορίθμους οδομετρίας και SLAM, δίνοντας βάση στα κοινά τους χαρακτηριστικά με τη δική μας υλοποίηση.

2.1 Πιθανοτικό SLAM

Το φίλτρο του *Bayes* αποτελεί τον πυρήνα της πιθανοτικής οδομετρίας καθώς αποτελεί τον πρώτο σχετικό αλγόριθμο που αναπτύχθηκε αλλά και την πηγή έμπνευσης για τις ακόλουθες πιθανοτικές υλοποιήσεις. Συγκεκριμένα, είναι ένας αναδρομικός αλγόριθμος ο οποίος προσδιορίζει την πόζα ενός ρομποτικού πράκτορα εκμεταλλευόμενος μία σειρά μετρήσεων και σημάτων ελέγχου. Ο αλγόριθμος του *Bayes* μπορεί να χωρισθεί σε δύο βήματα: το βήμα πρόβλεψης, και το βήμα ενημέρωσης της μέτρησης. Στο βήμα πρόβλεψης, η μέθοδος εφαρμόζει τα σήματα ελέγχου (όπως για παράδειγμα την ταχύτητα) έτσι ώστε να αποκτήσει μία αρχική πρόβλεψη για το διάνυσμα της πόζας του ρομποτικού πράκτορα. Στο βήμα ενημέρωσης της μέτρησης, χρησιμοποιώντας τις μετρήσεις από τους αισθητήρες ο αλγόριθμος διορθώνει κατάλληλα την πρότερη εκτίμηση της πόζας από το βήμα πρόβλεψης. Σύμφωνα με το [11] ο αλγόριθμος του *Bayes* μπορεί να ορισθεί ως εξής:

Algorithm 1 Αλγόριθμος Bayesian Filter [11]

Input $bel(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$
for all $\mathbf{x}_t$ **do**
  $\overline{bel}(\mathbf{x}_t) = \int \mathbb{P} = [\mathbf{x}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t]\ bel(\mathbf{x}_{t-1})\ \partial \mathbf{x}_{t-1}$
  $bel(\mathbf{x}_t) = \eta \cdot \mathbb{P} = [\mathbf{z}_t \mid \mathbf{x}_t]\ \overline{bel}(\mathbf{x}_t)$
end for
return $\mathrm{bel}(\mathbf{x}_t)$

Έχοντας ως πηγή έμπνευσης το φίλτρο *Bayes*, μεταγενέστερες μέθοδοι ξεκίνησαν σταδιακά να προσεγγίζουν το πρόβλημα του SLAM. Το *Extended Kalman* φίλτρο αποτελεί τη ναυαρχίδα και ιστορικά τον πρώτο αλγόριθμο για SLAM, ενώ μάλιστα αποτελεί πηγή έμπνευσης μέχρι και για σύγχρονες υλοποιήσεις. Πρακτικά είναι μια επέκταση του *Kalman* φίλτρου, το οποίο με τη σειρά του είναι μία μορφή του φίλτρου *Bayes* για δεδομένα που ακολουθούν την κανονική κατανομή και γραμμικά μοντέλα κίνησης και παρατήρησης, προσαρμοσμένο για τυχαίες κατανομές. Συγκεκριμένα, σύμφωνα με το [11], περιγράφεται ως εξής:

Algorithm 2 Αλγόριθμος Extended Kalman Filter για SLAM [11]

Input $(\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t,\ \mathbf{z}_t,\ N_{t-1})$

$N_t = N_{t-1}$

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 \end{pmatrix}$$

$$\overline{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{u_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{u_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ \frac{u_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ \omega_t\Delta_t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & \frac{u_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ 0 & 0 & \frac{u_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

$\overline{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + F_x^T\ R_t\ F_x$

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ **do**

$$\begin{pmatrix} \overline{\mu}_{N_t+1,x} \\ \overline{\mu}_{N_t+1,y} \\ \overline{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ sin(\phi_t^i + \overline{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

  for $k = 1$ to $N_t + 1$ **do**

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{k,x} - \overline{\mu}_{t,x} \\ \overline{\mu}_{k,y} - \overline{\mu}_{t,y} \end{pmatrix}$$

$q_k = \delta_k^T \delta_k$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \text{atan2}(\delta_{k,y}, \delta_{k,x}) - \overline{\mu}_{t,\theta} \\ \overline{\mu}_{k,s} \end{pmatrix}$$

$$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 & 0 & 0 & 0 & 0...0 \\ & & & & & & & \\ 0 & 0 & 0 & 0...0 & 1 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 1 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 0 & 1 & 0...0 \end{pmatrix}$$

$$H_t^k = \frac{1}{q_k} \begin{pmatrix} \sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & -\sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} & 0 \\ \delta_{k,y} & \delta_{k,x} & -1 & -\delta_{k,y} & -\delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$$

$\Psi_k = H_t^k\ \overline{\Sigma}_t\ [H_t^k]^T + Q_t$

$\pi_k = (z_t^i - \hat{z}_t^k)\ \Psi_k^{-1}\ (z_t^i - \hat{z}_t^k)$

  end for

$\pi_{N_t+1} = \alpha$

$j(i) = \arg\min_k \pi_k$

$N_t = \max(N_t, j(i))$

$K_t^i = \overline{\Sigma}_t\ [H_t^{j(i)}]^T\ \Psi_{j(i)}^{-1}$

end for

$\mu_t = \overline{\mu}_t + \sum_i K_t^i\ (z_t^i - \hat{z}_t^{j(i)})$

$\Sigma_t = (I - \sum_i K_t^i\ H_t^{j(i)})\ \overline{\Sigma}_t$

return $\mu_t,\ \Sigma_t$

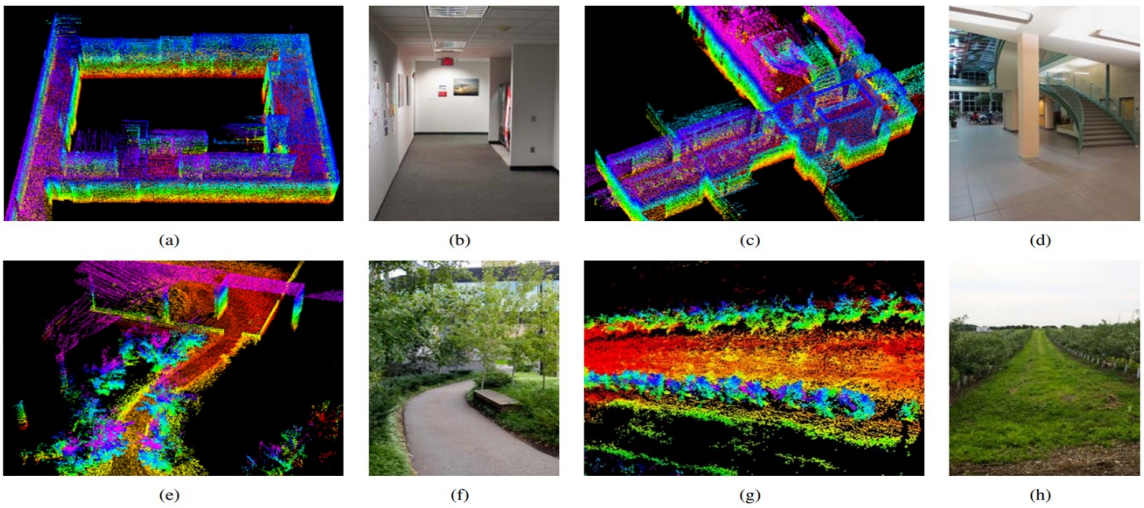
Όπου η μεταγενέστερη πιθανότητα *bel(*$\boldsymbol{x}_t$*)* τη χρονική στιγμή t περιγράφεται από τη μέση τιμή μ_t και τη συνδιακύμανση Σ_t, και R_t, Q_t είναι οι πίνακες συνδιακύμανσης σχετικά με τα ε_t και δ_t, αντίστοιχα. Όπως το φίλτρο *Bayes*, έτσι και το *Extended Kalman* φίλτρο περιλαμβάνει τα βήματα της πρόβλεψης και ανανέωσης μέτρησης τα οποία είναι διακριτά στα βήματα 2 με 5, και 23 με 24, αντίστοιχα. Μία σημαντική διαφορά των δύο αλγορίθμων είναι η παρουσία του *Kalman Gain*, το οποίο αποτυπώνεται στο βήμα 21 και σταθμίζει το πόσο θα επηρεάσει την τελική εκτίμηση το βήμα μέτρησης. Ακόλουθα από το *Extended Kalman* φίλτρο, ακόμα περισσότερες πιθανοτικές υλοποιήσεις κέντρισαν το ενδιαφέρον της επιστημονικής κοινότητας, με επικρατούσα το *Rao-Blackwellised Particle* φίλτρο. Το φίλτρο αυτό ενσωματώνει το θεώρημα Rao–Blackwell στον κορμό του *Particle* φίλτρου ώστε να βελτιώσει τη διαδικασία της δειγματοληψίας. Οι πρώτες δουλειές σχετικά με τον αλγόριθμο αυτό περιγράφονται στα [16] και [17].

2.2 LiDAR SLAM

Το LiDAR SLAM αποτελεί μία από τις κύριες κατηγορίες του SLAM τα τελευταία χρόνια, με τον LOAM (Lidar Odometry and Mapping) [2] να είναι ο πιο δημοφιλής αλγόριθμος που το αντιπροσωπεύει. Ο πυρήνας του LOAM έχει αρκετές ομοιότητες με τις μεθόδους του οπτικού SLAM που βασίζονται σε χαρακτηριστικά της εικόνας, με την κύρια να είναι ότι και αυτός στις βελτιστοποιήσεις ελαχιστοποιεί την απόσταση μεταξύ χαρακτηριστικών που έχουν εξαχθεί από το LiDAR. Οι δύο επιμέρους αλγόριθμοι του LOAM σχετικά με τον προσδιορισμό της πόζας και την χαρτογράφηση τρέχουν σε διαφορετικές συχνότητες. Συγκεκριμένα, ο πρώτος λειτουργεί στη συχνότητα των 10Hz, ενώ ο δεύτερος μόνο μία φορά στο τέλος κάθε πλήρους σάρωσης (1Hz). Αναφορικά με τον πρώτο, κατά τη σάρωση, για να προσδιοριστούν οι αντιστοιχίες σημείων μεταξύ της τρέχουσας και της προηγούμενης σάρωσης, τα σημεία υπόκεινται σε ένα φιλτράρισμα τριών επιπέδων και χωρίζονται κατάλληλα σε edge και planar σημεία ώστε να ζευγοποιηθούν πιο αποδοτικά. Από τη στιγμή που θα προσδιοριστεί μία αντιστοιχία, αν αυτή επιβιώσει από έναν διπλό έλεγχο που γίνεται σχετικά με τη θέση των σημείων, θα εισαχθεί στη διαδικασία της βελτιστοποίησης. Από την άλλη μεριά, πάλι κατατάσσοντας τα σημεία σε edge και planar, ο αλγόριθμος χαρτογράφησης βελτιστοποιεί ξανά την πόζα του LiDAR στο τέλος κάθε πλήρους σάρωσης εντοπίζοντας αντιστοιχίσεις μεταξύ αυτής και του χάρτη. Στην εικόνα 1 απεικονίζονται μερικές ανακατασκευές του περιβάλλοντα χώρου μέσω του LOAM.

2.3 Οπτικό SLAM

Η υπέρμετρη ανάγκη για αυτονομία στις σύγχρονες εφαρμογές σε συνδυασμό με την μεγάλη διαθεσιμότητα σε χαμηλού κόστους κάμερες οι οποίες είναι εξαιρετικά αποδοτικές όσον αφορά το SLAM, έχουν οδηγήσει στην ανάπτυξη πληθώρας πρωτοποριακών αλγορίθμων οπτικού SLAM. Ξεκινώντας από πιθανοτικές και προχωρώντας σε άμεσες (direct)



Εικόνα 1: Χάρτες οι οποίοι έχουν εξαχθεί από (a)-(b) έναν στενό και έναν φαρδύ διάδρομο, (c)-(d) μία μεγάλη αίθουσα, (e)-(f) έναν δρόμο με βλάστηση, and (g)-(h) έναν δρόμο ανάμεσα από δύο σειρές δέντρων [2].

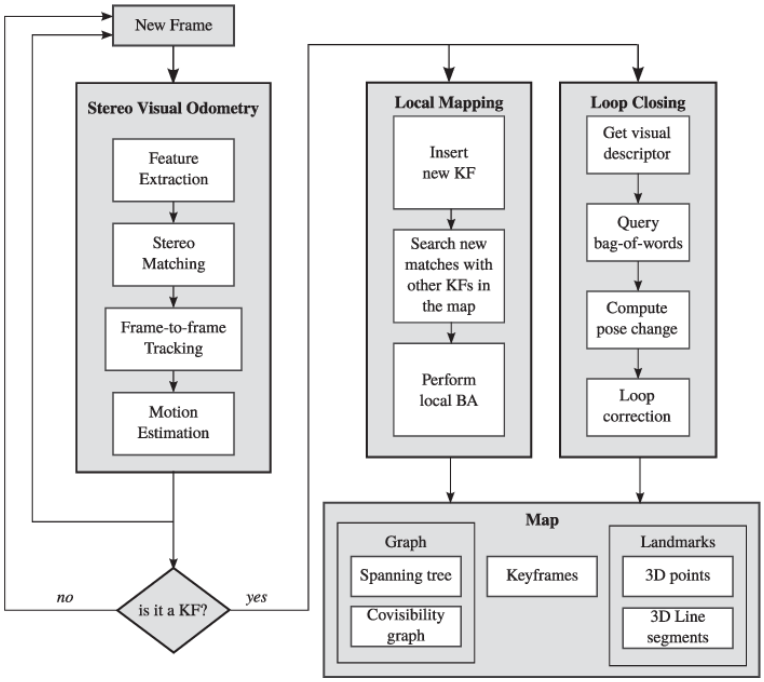
μεθόδους, στο επίκαιρο SLAM χρησιμοποιούνται κατά κύριο λόγο αλγόριθμοι που βασίζονται σε χαρακτηριστικά της εικόνας τα οποία μάλιστα πλέον ποικίλλουν και μπορούν να καταταχθούν σε κατηγορίες χαμηλού επιπέδου όπως είναι τα γεωμετρικά χαρακτηριστικά (παραδείγματος χάριν σημεία και ευθύγραμμα τμήματα), αλλά και υψηλού επιπέδου όπως για παράδειγμα τα σημασιολογικών επισημειωμένα αντικείμενα. Μερικοί από τους βασικότερους αλγορίθμους οπτικού SLAM παρουσιάζονται περιληπτικά στον πίνακα 1.

Μία από τις πιο αξιόλογες υλοποιήσεις SLAM με ευθείες και γραμμές είναι το stereo PL-SLAM [3], το οποίο μάλιστα ενέπνευσε σε μεγάλο βαθμό την εργασία μας. Το PL-SLAM μπορεί να χωριστεί σε τρία επιμέρους κομμάτια: την οπτική οδομετρία (visual odometry), την τοπική χαρτογράφηση (local mapping), και το κλείσιμο βρόχου (loop closing) (εικόνα 2). Σχετικά με το κομμάτι της οπτικής οδομετρίας, η μέθοδος εκτιμά την πόζα της κάμερας προσδιορίζοντας αντιστοιχίες χαρακτηριστικών μεταξύ της τρέχουσας και προηγούμενης εικόνας. Προκειμένου να ταυτοποιηθεί μία ζευγοποίηση γραμμών, επιβεβαιώνεται από τον αλγόριθμο ότι αυτές είναι αμφίδρομα οι καλύτερες υποψήφιοι μεταξύ τους, μια πολιτική την οποία ακολουθήσαμε και εμείς στην υλοποίηση μας. Για την επιλογή εικόνων-κλειδιά, η μέθοδος αυτή υιοθετεί την ιδέα ότι θα πρέπει να εισάγονται με τέτοιο τρόπο ώστε η κάμερα να βρίσκεται πάντα σχετικά κοντά σε μία εικόνα κλειδί (KF) και επομένως να μην συσσωρεύεται σφάλμα. Συγκεκριμένα, ακολουθώντας την μεθοδολογία που επεξηγείται στο [65], μία εικόνα-κλειδί εισάγεται μόνο όταν το πηλίκο της εντροπίας της εκτίμησης της κίνησης μεταξύ της προηγούμενης εικόνας-κλειδί και της τρέχουσας εικόνας προς την

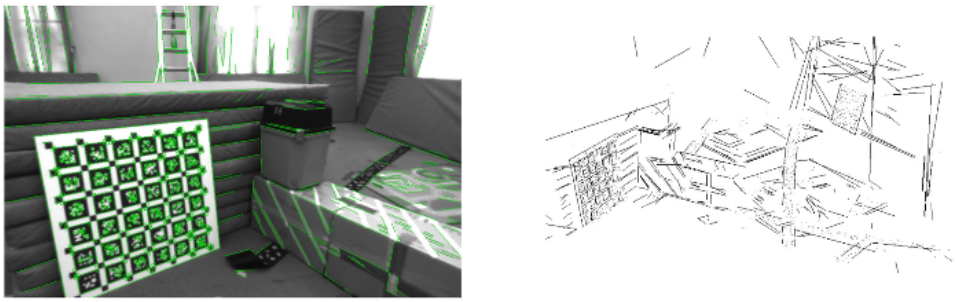
Πίνακας 1: Σύνοψη των πιο αντιπροσωπευτικών συστημάτων οπτικού SLAM και οδομετρίας.

Σύστημα	**SLAM ή Οδομετρία**	**Εντοπισμός Χαρακτηριστικών**	**Συσχέτιση Δεδομένων**	**Μοντέλο Κάμερας**
Mono-SLAM [18, 19, 20]	SLAM	Shi Tomasi	Correlation	Monocular
PTAM [21, 22, 23, 24]	SLAM	FAST	Pyramid SSD	Monocular
LSD-SLAM [25, 26, 27]	SLAM	Edgelets	Direct	Monocular, Stereo
SVO [28, 29, 30]	VO	FAST+Hi grad.	Direct	Monocular, Stereo, Fisheye
ORB-SLAM2 [31, 32]	SLAM	ORB	Descriptor	Monocular, Stereo
DSO [33, 34, 35, 36]	VO	High grad.	Direct	Monocular, Stereo, Fisheye
DSM [37, 38]	SLAM	High grad.	Direct	Monocular
PL-SLAM [13, 39]	SLAM	ORB+LSD	Descriptor	Monocular
PL-SLAM [3, 40]	SLAM	ORB+LSD	Descriptor	Stereo
PL-VIO [14, 41]	SLAM	ORB+LSD	Descriptor	Monocular + IMU
PL-VINS [42, 43]	SLAM	ORB+LSD	Descriptor	Monocular + IMU
MSCKF [44, 45, 46, 47]	VO	Shi Tomasi	Cross Correlation	Monocular, Stereo + IMU
OKVIS [48, 49, 50]	VO	BRISK	Descriptor	Monocular, Stereo, Fisheye + IMU
ROVIO [51, 52, 53]	VO	Shi Tomasi	Direct	Monocular, Stereo, Fisheye + IMU
ORBSLAM-VI [54]	SLAM	ORB	Descriptor	Monocular + IMU
VINS-Fusion [55, 56, 57]	VO	Shi Tomasi	KLT	Monocular, Stereo, Fisheye + IMU
VI-DSO [58]	VO	High grad.	Direct	Monocular + IMU
BASALT [59, 60]	VO	FAST	KLT (LSSD)	Stereo, Fisheye + IMU
Kimera [61, 62]	VO	Shi Tomasi	KLT	Stereo + IMU
ORB-SLAM3 [1, 63]	SLAM	ORB	Descriptor	Monocular, Stereo, Fisheye + IMU
ORB-LINE-SLAM [64] (δικό μας)	SLAM	ORB+LSD or EDLines	Descriptor	Stereo + IMU

εντροπία της κίνησης μεταξύ της προηγούμενης εικόνας-κλειδί και της αμέσως επόμενης από αυτήν εικόνας, είναι μικρότερο από το πειραματικά ορισμένο κατώφλι με τιμή 0.9. Στο κομμάτι της τοπικής χαρτογράφησης, ο αλγόριθμος ουσιαστικά βελτιστοποιεί την πόζα της κάμερας αλλά και χαρακτηριστικών του χάρτη μέσα από μία ομάδα εικόνων οι οποίες μοιράζονται κοινά χαρακτηριστικά, ξανά μέσω της διαδικασίας του bundle adjustment το οποίο χρησιμοποιείται όμοια και στο πρώτο κομμάτι. Τέλος, μέσω ενός ειδικού τύπου λεξιλογίου ο αλγόριθμος έχει τη δυνατότητα να μεταφράζει την οπτική πληροφορία σε διανύσματα ώστε να επιταχύνεται ο εντοπισμός βρόχου, μέσω του κλεισίματος του οποίου διορθώνεται δραστικά το σφάλμα της πόζας το οποίο ολοκληρώνεται διαρκώς καθώς τρέχει ο αλγόριθμος.



Εικόνα 2: Δομή του PL-SLAM [3].

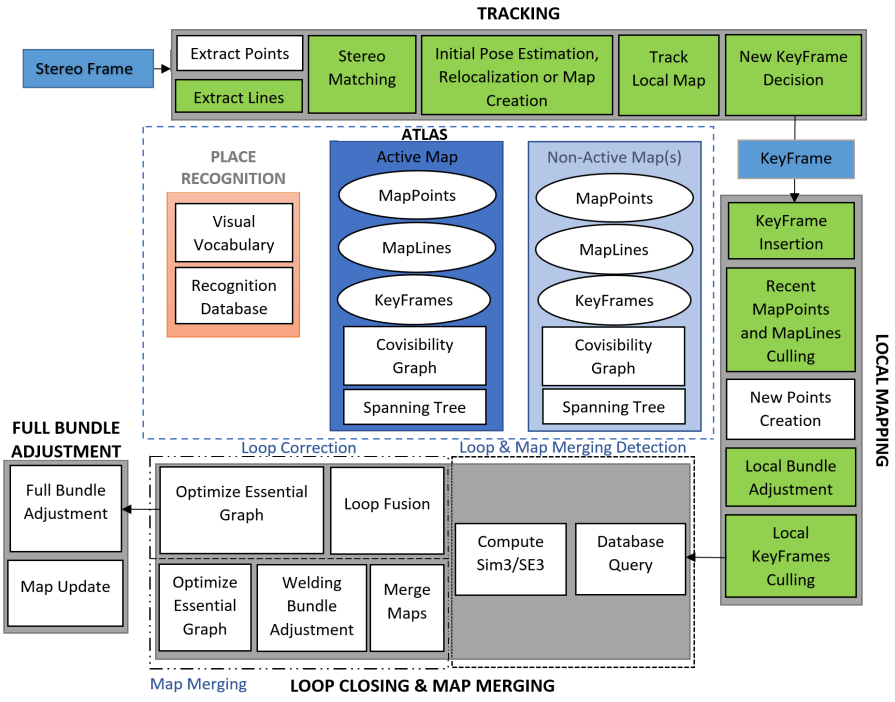


Εικόνα 3: Ανακατασκευή χάρτη μέσω του PL-SLAM [3].

3. Περιγραφή του ORB-LINE-SLAM Συστήματος

3.1 Εισαγωγή

Η κεντρική ιδέα πίσω από την εργασία μας ήταν να δημιουργήσουμε ένα υβριδικό σύστημα, όπου ο κορμός του θα αποτελείται από έναν οπτικό SLAM αλγόριθμο που θα χρησιμοποιεί ταυτόχρονα ευθείες και γραμμές, αλλά στον οποίο οι γραμμές θα έχουν βοηθητικό ρόλο ως προς τα σημεία έτσι ώστε να ενισχυθεί η ανθεκτικότητα και αποδοτικότητα του αλγορίθμου κάτω από απαιτητικές συνθήκες. Την ίδια στιγμή, είχαμε και το κίνητρο να αναπτύξουμε και έναν νέο αλγόριθμο που θα βασίζεται μόνο σε γραμμές και θα έχει τη δυνατότητα να λειτουργεί ικανοποιητικά σε πραγματικές συνθήκες χωρίς καμία συμμετοχή των σημείων. Από τους δύο παραπάνω στόχους προέκυψε το υβριδικό σύστημα ORB-LINE-SLAM το οποίο σε διαφορετικές λειτουργίες μπορεί να χρησιμοποιεί είτε ταυτόχρονα σημεία και γραμμές, ή αποκλειστικά γραμμές. Η υλοποίησή μας μπορεί να διαιρεθεί σε τρεις φάσεις: τη φάση της παρακολούθησης (tracking), της τοπικής χαρτογράφησης (local mapping), και του κλεισίματος βρόχου (loop closing). Οι γραμμές συμμετέχουν πλήρως στις πρώτες δύο φάσεις, ενώ στην τρίτη απλά διορθώνεται η θέση τους καθώς κλείνει ο βρόχος και οπότε η σύνθεση παραμένει η ίδια με του ORB-SLAM3 [1]. Περιληπτικά η δομή του συστήματος περιγράφεται στην εικόνα 4.



Εικόνα 4: Συνοπτική απεικόνιση της δομής του ORB-LINE-SLAM. Τα τμήματα με πράσινο χρώμα είναι αυτά στα οποία συμπεριλαμβάνονται οι γραμμές.

3.2 Στάδιο Παρακολούθησης

Ο πρώτος μας στόχος κατά το σχηματισμό του συστήματος ήταν η επιλογή κατάλληλου αλγορίθμου για τον εντοπισμό των ευθυγράμμων τμημάτων. Για να επιλύσουμε το πρόβλημα αυτό, στην υλοποίησή μας συμπεριλάβαμε τους δύο πιο γνωστούς αλγορίθμους και τους συγκρίναμε σχετικά με την απόδοσή τους στο πρόβλημα του SLAM. Αυτοί είναι ο Line Segment Detector (LSD) [66] και ο EDlines[67]. Σύμφωνα με τα πειράματά μας, ο πρώτος κυριαρχεί σημαντικά σε όρους απόδοσης λόγω του ότι εμφανίζει πολύ μεγαλύτερη επαναληψιμότητα στο να εντοπίζει τα ίδια ευθύγραμμα τμήματα σε διαφορετικές εικόνες, και επομένως να προσδιορίζει εξαιρετικά περισσότερες αντιστοιχίες. Για τον λόγο αυτό, για την εκτίμηση της απόδοσης του συστήματος (κεφάλαιο 5) χρησιμοποιήσαμε τον LSD. Παρόλα αυτά, ο EDlines είναι μακράν ταχύτερος, χαρακτηριστικό το οποίο είναι πολύ κρίσιμο για την απόδοση της μεθόδου σε πραγματικό χρόνο. Χρησιμοποιώντας όμως έναν προσαρμοσμένο LSD μέσω της OpenCV βιβλιοθήκης [68], μας δόθηκε η δυνατότητα να μειώσουμε την αρχική κλίμακα της εικόνας ελαττώνοντας κατακόρυφα τον απαραίτητο υπολογιστικό χρόνο. Για παράδειγμα, μέχρι και με αρχική κλίμακα ίση με 0.5 ο LSD είναι αποδοτικότερος, ενώ απαιτεί περίπου τον ίδιο χρόνο με τον EDlines. Ένας ακόμη παράγοντας κλειδί της υλοποίησής μας σχετικά με την εξαγωγή γραμμών είναι ότι τα ευθύγραμμα τμήματα εντοπίζονται από δύο επίπεδα κλίμακας της εικόνας, γεγονός το οποίο είναι εξαιρετικά κρίσιμο αν σκεφτεί κανείς ότι ο αριθμός των αντιστοιχιών είναι αρκετές φορές οριακά επαρκής για να διεξαχθούν με ασφάλεια οι βελτιστοποιήσεις. Ως προς τον descriptor, επιλέξαμε τη συνήθη επιλογή για το πεδίο του SLAM με γραμμές, όπου είναι ο 256 bits Line Band Descriptor (LBD) [69] ο οποίος αναπαριστά κάθε ευθύγραμμο τμήμα σύμφωνα με την εμφάνιση του τοπικού του χώρου.

Για την ασφαλή εκτίμηση της πόζας της κάμερας, το σύστημα πρώτα πρέπει να διασφαλίσει ότι θα υπάρχει επαρκής αριθμός εισόδων στην διαδικασία της βελτιστοποίησης. Για τον λόγο αυτό, για την αρχικοποίηση του συστήματος η μέθοδος απαιτεί ότι στο αρχικό frame θα πρέπει να εντοπίζονται συνολικά τουλάχιστον 500 ORBS και ευθύγραμμα τμήματα έτσι ώστε στη συνέχεια να είναι βέβαιο πως θα προσδιοριστεί ένας ικανοποιητικός αριθμός αντιστοιχιών. Αν το σύστημα αποτύχει να εντοπίσει αυτό τον αριθμό χαρακτηριστικών, σταματάει, και επαναλαμβάνει την ίδια διαδικασία στην επόμενη εικόνα. Εν συνεχεία, για να γίνει η πρόβλεψη της πόζας ο αλγόριθμος αρχικά εντοπίζει αντιστοιχίες μεταξύ της τρέχουσας και της προηγούμενης εικόνας και μέσω του bundle adjustment το οποίο αφορά μόνο την κίνηση της κάμερας και περιγράφεται λεπτομερώς στο κεφάλαιο 4, υπολογίζει την πρώτη εκτίμηση. Όμοια με τη διαδικασία αυτή, ακολούθως, το σύστημα προσδιορίζει ζευγάρια χαρακτηριστικών μεταξύ εικόνων-κλειδιά (keyframes) που μοιράζονται μεταξύ τους κοινά χαρακτηριστικά και ορίζουν έναν τοπικό χάρτη, και βελτιστοποιεί αποκλειστικά την πόζα ξανά μέσω του bundle adjustment. Για να οριστεί μια εικόνα ως εικόνα-κλειδί ακολουθείται όμοια τακτική με το ORB-SLAM2 [31], η οποία αποφασίζει ανάλογα με τον αριθμό των κοντινών και μακρινών χαρακτηριστικών. Συγκεκριμένα, ορίζοντας ως κατώφλι ~ 40 φορές το

stereo baseline, όπως προτείνεται στο [70], ο αλγόριθμος κατατάσσει μία εικόνα ως εικόνα-κλειδί αν και μόνο αν λιγότερα από 140 κοντινά χαρακτηριστικά της χρησιμοποιήθηκαν στη βελτιστοποίηση και δημιουργήθηκαν τουλάχιστον 100 νέα κοντινά. Τα αντίστοιχα κατώφλια για τον αλγόριθμο μόνο με γραμμές είναι 40 και 30. Αυτή η πολιτική απορρέει από το γεγονός ότι τα κοντινά χαρακτηριστικά αποδίδουν πλούσια πληροφορία για την μετατόπιση και την περιστροφή, ενώ τα μακρινά μόνο για την περιστροφή. Ένα ευθύγραμμο τμήμα, για να καταταχθεί ως κοντινό, θα πρέπει το βάθος και των δύο ακραίων του σημείων να μην ξεπερνάει το προαναφερθέν κατώφλι.

3.3 Τοπική Χαρτογράφηση

Αφότου εισαχθεί μία νέα εικόνα-κλειδί, το σύστημα ακολουθεί μία αυστηρή πολιτική για την περικοπή των πλεοναζόντων σημείων και γραμμών που έχουν πιθανώς προκύψει από αστοχίες στην αντιστοίχιση και νοθεύουν τις διαδικασίες βελτιστοποίησης αλλά και ανακατασκευής του χάρτη. Οι συνθήκες που θα πρέπει να ικανοποιούν οι γραμμές είναι:

1. Η γραμμή θα πρέπει να εντοπίζεται σε περισσότερες από το 20% των εικόνων στις οποίες είναι ορατή μέσα από τον τοπικό χάρτη.

2. Εάν περισσότερες από μία εικόνες-κλειδιά έχουν περάσει από τη δημιουργία της γραμμής, θα πρέπει να παρατηρείται από τουλάχιστον δύο εικόνες-κλειδιά.

Στη συνέχεια το σύστημα δημιουργεί νέα σημεία μέσω του τριγωνισμού σημείων που δεν έχουν αντιστοιχιστεί ως εκείνη τη στιγμή και παρατηρούνται σε εικόνες-κλειδιά που συνδέονται στο γράφο. Η διαδικασία αυτή ήταν αρκετά απαιτητικό να επεκταθεί και στις γραμμές καθώς τα ακραία τους σημεία είναι αρκετά ασταθή ειδικά όταν η ίδια γραμμή εντοπίζεται υπό διαφορετικές οπτικές γωνίες. Παρόλα αυτά, για το συγκεκριμένο θέμα απαιτείται και προτείνεται περαιτέρω έρευνα. Η διαδικασία του τοπικού bundle adjustment που ακολουθεί, η οποία είναι και η κύρια για το στάδιο αυτό, περιγράφεται αναλυτικά στο επόμενο κεφάλαιο. Η τελευταία διαδικασία της τοπικής χαρτογράφησης είναι η περικοπή των περιττών τοπικών εικόνων-κλειδιά τα οποία επιβαρύνουν τη μνήμη του συστήματος αλλά και τις διαδικασίες της βελτιστοποίησης χωρίς να εμπεριέχουν καμία επιπρόσθετη σημαντική πληροφορία. Συγκεκριμένα, μια εικόνα-κλειδί αποκόπτεται αν τουλάχιστον το 90% των χαρακτηριστικών της είναι ορατό σε τουλάχιστον τρεις ακόμα εικόνες-κλειδιά στην ίδια ή καλύτερη κλίμακα. Με τον τρόπο αυτό, πέρα από τα παραπάνω, η μέθοδος δημιουργεί μία πολύ πιο συνεκτική αναπαράσταση του περιβάλλοντος χώρου καθώς δεν περιλαμβάνει εικόνες-κλειδιά που περιέχουν σε μεγάλο βαθμό παρόμοια πληροφορία.

4. Bundle Adjustment

4.1 Συναρτήσεις Σφάλματος για τα Ευθύγραμμα Τμήματα

Με σκοπό να διαλέξουμε την πιο αποδοτική συνάρτηση σφάλματος για τα ευθύγραμμα τμήματα, υιοθετήσαμε στο σύστημα μας και συγκρίναμε τις πιο ευρέως διαδεδομένες που εντοπίσαμε στην βιβλιογραφία. Αν ο $\Xi_{iw} \in \mathrm{SE}(3)$ αποτυπώνει την πόζα της κάμερας στην i-οστή εικόνα, $P_{w,k}$, $Q_{w,k} \in \mathbb{R}^3$ είναι τα τρισδιάστατα ακραία σημεία του k-ιοστού ευθύγραμμου τμήματος που παρατηρείται στην i-οστή εικόνα, και $p_{i,k}, q_{i,k} \in \mathbb{R}^2$ τα δισδιάστατα ακραία σημεία στο επίπεδο της εικόνας που αντιστοιχούν στα ακραία σημεία του k-ιοστού ευθύγραμμου τμήματος στην i-οστή εικόνα, όπου w το σημείο αναφοράς του συστήματος συντεταγμένων του κόσμου. Οι τρεις συναρτήσεις σφάλματος που χρησιμοποιούνται στο σύστημά μας ορίζονται ως εξής:

1. Η πρώτη συνάρτηση σφάλματος, όπως παρουσιάζεται στο [3], χρησιμοποιεί την Ευκλείδεια απόσταση μεταξύ των ακραίων σημείων του επαναπροβαλλόμενου τρισδιάστατου ευθύγραμμου τμήματος και της ευθείας που ορίζει το αντιστοιχισμένο σε αυτό ευθύγραμμο τμήμα, στο επίπεδο της εικόνας. Η εξίσωση της ευθείας μπορεί να υπολογιστεί ως το εξωτερικό γινόμενο μεταξύ των $p_{i,k}$ και $q_{i,k}$, και συγκεκριμένα ορίζεται ως:

$$l_{i,k} = \frac{{}^h p_{i,k} \times {}^h q_{i,k}}{\| {}^h p_{i,k} \times {}^h q_{i,k} \|}, \tag{1}$$

όπου ${}^h p_{i,k}, {}^h p_{i,k} \in \mathbb{R}^3$ αντιστοιχούν στις ομογενείς συντεταγμένες των $p_{i,k}, q_{i,k}$. Επομένως η συνάρτηση σφάλματος αποτυπώνεται ως εξής:

$$e_{i,k} = \begin{bmatrix} l_{i,k} \cdot {}^h\pi(\Xi_{iw}, P_{w,k}) \\ l_{i,k} \cdot {}^h\pi(\Xi_{iw}, Q_{w,k}) \end{bmatrix}, \tag{2}$$

όπου ${}^h\pi(\Xi_{iw}, P_{w,k})$, ${}^h\pi(\Xi_{iw}, Q_{w,k})$ είναι οι ομογενείς συντεταγμένες των $\pi(\Xi_{iw}, P_{w,k})$, $\pi(\Xi_{iw}, Q_{w,k})$, αντίστοιχα, οι οποίες αναπαριστούν τα δισδιάστατα ακραία σημεία του επαναπροβαλλόμενου ευθύγραμμου τμήματος στο επίπεδο της εικόνας και ορίζονται ως:

$$\pi(\Xi_{iw}, P_{w,k}) = \begin{bmatrix} f_x \dfrac{m_x}{m_z} + c_x \\ f_y \dfrac{m_y}{m_z} + c_y \end{bmatrix}, \tag{3}$$

$$m = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T = R_{iw} P_{w,k} + t_{iw}, \tag{4}$$

όπου $R_{iw} \in \mathrm{S0}(3), t_{iw} \in \mathbb{R}^3$ είναι οι πίνακες περιστροφής και μετατόπισης αναφορικά με τον Ξ_{iw}, (f_x, f_y) το εστιακό μήκος, και (c_x, c_y) το κύριο σημείο (principal point), και τα δύο γνωστά από τη βαθμονόμηση της κάμερας. Όμοια με τις εξισώσεις 3 και 4, το $\pi(\Xi_{iw}, Q_{w,k})$

μπορεί εύκολα να υπολογιστεί. Κατά τη διαδικασία του bundle adjustment, το σφάλμα επαναπροβολής αναφορικά με ένα ζευγάρι ταιριασμένων γραμμών φιλτράρεται ως έκπτωτη τιμή αν και μόνο αν $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 7.8$, όπου $\Sigma_{e_{i,k}}$ είναι ο μοναδιαίος πίνακας που αναφέρεται στην συνδιακύμανση. Ως σύνηθες στο πεδίο του SLAM, το κατώφλι αυτό προσδιορίστηκε πειραματικά, καθώς συγκεκριμένα μέσω αυτού πήραμε τα καλύτερα αποτελέσματα στα πειράματα μας.

2. Η δεύτερη συνάρτηση σφάλματος, όπως παρουσιάζεται στο [71], χρησιμοποιεί την απόσταση γωνίας μεταξύ της παρατηρηθείσας και της επαναπροβαλλόμενης γραμμής. Υποθέτοντας τις προηγούμενες παραδοχές, και αν $p'_{i,k}, q'_{i,k} \in \mathbb{R}^2$ αναπαριστά τα διδιάστατα ακραία σημεία του επαναπροβαλλόμενου ευθύγραμμου τμήματος στο επίπεδο της εικόνας, ακριβώς όπως τα $\pi(\Xi_{iw}, P_{w,k}), \pi(\Xi_{iw}, Q_{w,k})$, τότε η συνάρτηση σφάλματος αποτυπώνεται λεπτομερώς ως εξής:

$$e_{i,k} = \begin{bmatrix} \dfrac{q_{i,k}\vec{p}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|q_{i,k}\vec{p}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} - 1 \\ \dfrac{p_{i,k}\vec{q}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|p_{i,k}\vec{q}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} + 1 \end{bmatrix}. \tag{5}$$

Η συνάρτηση σφάλματος αναπαριστά το συνημίτονο μεταξύ των διανυσμάτων $q_{i,k}\vec{p}'_{i,k}, q_{i,k}\vec{p}_{i,k}$, και $p_{i,k}\vec{q}'_{i,k}, q_{i,k}\vec{p}_{i,k}$. Όμοια με πριν, το σφάλμα ενός ζεύγους ευθύγραμμων τμημάτων θεωρείται ως έκπτωτη τιμή αν $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 7 \cdot 10^{-5}$, όπου $\Sigma_{e_{i,k}}$ είναι ξανά ο μοναδιαίος πίνακας. Το κατώφλι αυτό τέθηκε και πάλι πειραματικά.

3. Τέλος, συνδυάζουμε τις δύο προηγούμενες συναρτήσεις, και προκύπτει μία συνολική συνάρτηση που εμπεριέχει όλα τα σφάλματα που περιγράψαμε προηγουμένως (εικόνα 5), και περιγράφεται από την εξίσωση:

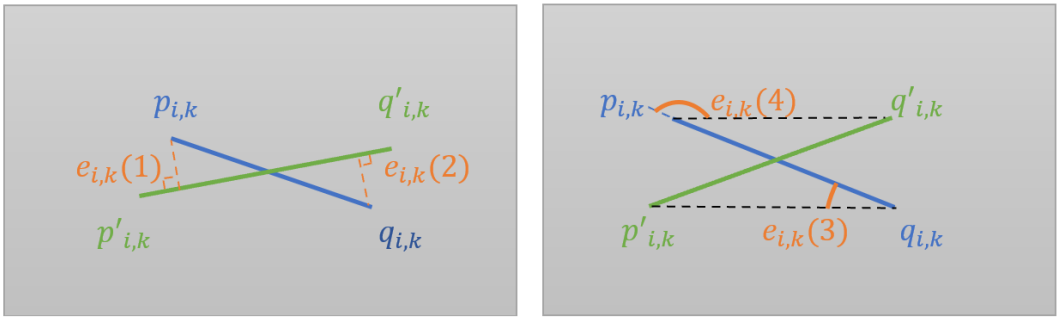
$$e_{i,k} = \begin{bmatrix} l_{i,k} \cdot {}^h\pi(\Xi_{iw}, P_{w,k}) \\ l_{i,k} \cdot {}^h\pi(\Xi_{iw}, Q_{w,k}) \\ \dfrac{q_{i,k}\vec{p}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|q_{i,k}\vec{p}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} - 1 \\ \dfrac{p_{i,k}\vec{q}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|p_{i,k}\vec{q}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} + 1 \end{bmatrix}. \tag{6}$$

Όσον αφορά αυτή τη συνάρτηση σφάλματος, θέτουμε σε εφαρμογή μία ελαφρώς διαφορετική στρατηγική για να εντοπίζουμε έκπτωτες τιμές. Αναλυτικά, το κύριο πρόβλημα είναι ότι οι δύο επιμέρους συναρτήσεις εκφράζουν διαφορετικές αποστάσεις και επομένως οι τιμές τους δεν είναι στην ίδια τάξη μεγέθους. Για να αντιμετωπίσουμε το πρόβλημα αυτό, χρησιμοποιώντας τον αντίστροφο πίνακα της συνδιακύμανσης που στην ουσία παίζει ρόλο πίνακα στάθμισης, κανονικοποιούμε την τιμή κάθε παράγοντα της συνάρτησης σφάλματος

σύμφωνα με τα προηγουμένως ορισθέντα κατώφλια, έτσι ώστε κάθε επιμέρους σφάλμα να φτάνει στην τιμή 1 όταν είναι έκπτωτη τιμή, σε αντιστοιχία με τις προηγούμενες περιπτώσεις. Συγκεκριμένα, ο αντίστροφος πίνακας της συνδιακύμανσης ορίζεται:

$$\Sigma_{e_{i,k}}^{-1} = \begin{bmatrix} \frac{1}{7.8} & 0 & 0 & 0 \\ 0 & \frac{1}{7.8} & 0 & 0 \\ 0 & 0 & \frac{1}{7 \cdot 10^{-5}} & 0 \\ 0 & 0 & 0 & \frac{1}{7 \cdot 10^{-5}} \end{bmatrix}. \tag{7}$$

Με τον τρόπο αυτό, όχι μόνο μπορούμε να απορρίπτουμε αποδοτικότερα έκπτωτες τιμές χρησιμοποιώντας πληροφορία που αφορά και την Ευκλείδεια απόσταση και τη γωνία, αλλά επίσης σταθμίζονται κατάλληλα και οι δύο συναρτήσεις ώστε να συνεισφέρουν σχεδόν εξίσου στη διαδικασία του bundle adjustment. Το κατώφλι για τις έκπτωτες τιμές τέθηκε πειραματικά στην τιμή 1.5 (δηλαδή, $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 1.5$).



Εικόνα 5: Σφάλματα επαναπροβολλής για τα ευθύγραμμα τμήματα. Η αριστερή εικόνα αποτυπώνει την Ευκλείδεια απόσταση ενώ η δεξιά την απόσταση γωνίας.

Όπως θα αναλυθεί εκτενώς στο επόμενο κεφάλαιο, σύμφωνα με τα πειράματα μας, η πιο αποδοτική συνάρτηση σφάλματος είναι αυτή που λαμβάνει υπόψιν μόνο την Ευκλείδεια απόσταση, και για το λόγο αυτό, αυτή είναι και η συνάρτηση που επιλέξαμε για να αξιολογήσουμε το σύστημα μας που χρησιμοποιεί ταυτόχρονα σημεία και γραμμές.

4.2 Αποκλειστικά Βάσει Κίνησης και Τοπικό Bundle Adjustment

Το σύστημα εκτελεί bundle adjustment που αφορά μόνο στην κίνηση της κάμερας στην ενότητα της παρακολούθησης ώστε να βελτιστοποιήσει μόνο την πόζα της κάμερας, ενώ αντίθετα στην ενότητα της τοπικής χαρτογράφησης πραγματοποιεί τοπικό bundle adjustment κατά το οποίο βελτιστοποιεί την πόζα της κάμερας σε μία ομάδα εικόνων αλλά και

τη θέση χαρακτηριστικών στο χάρτη. Και στις δύο περιπτώσεις, για τις βελτιστοποιήσεις χρησιμοποιείται η μέθοδος Levenberg-Marquardt από τη βιβλιοθήκη g2o [72].

Για τον αποδοτικό συνδυασμό των σημείων και γραμμών στο bundle adjustment, προτείνουμε έναν νέο προσαρμοστικό παράγοντα ο οποίος σταθμίζει τη συμμετοχή των γραμμών κατά τη διαδικασία. Η σκέψη αυτή προέκυψε από το γεγονός ότι κατά τη μορφοποίηση της συνάρτησης σφάλματος (Ευκλείδεια απόσταση) σχετικά με τις γραμμές λανθασμένα υποθέτουμε ότι τα ακραία σημεία της επαναπροβαλλόμενης γραμμής αντιστοιχούν με τα κοντινότερα με αυτά σημεία στην ευθεία που ορίζεται από την παρατηρούμενη γραμμή στο επίπεδο της εικόνας. Στην πραγματικότητα, τα ακραία σημεία κάθε ευθύγραμμου τμήματος είναι ασταθή και δεν μπορούν να επαναπροσδιορισθούν με ακρίβεια από εικόνα σε εικόνα. Λόγω της παραδοχής αυτής, οι γραμμές είναι συνυφασμένες με μεγαλύτερο σφάλμα απ' ότι τα σημεία. Παρόλα αυτά, υπάρχουν περιπτώσεις όπου τα σημεία δεν επαρκούν για να γίνουν με ασφαλή τρόπο οι βελτιστοποιήσεις. Σε τέτοιες περιπτώσεις, είτε το σύστημα χάνει τη θέση της κάμερας, ή η εκτίμηση συχνά παραβιάζεται σημαντικά παρατηρώντας σημαντικές μεταπηδήσεις στον υπολογισμό της θέσης λόγω αστοχιών στην αντιστοίχιση σημείων. Από τα παραπάνω λοιπόν, είναι προφανές ότι οι γραμμές θα ήταν χρήσιμες όταν υπάρχει μικρός αριθμός αντιστοιχίσεων σημείων, και το αντίστροφο. Ακολουθώντας τη φιλοσοφία αυτή, ο προσαρμοστικός παράγοντας σταθμίζει ανάλογα τη συνεισφορά των γραμμών στον αλγόριθμο.

1. *Αποκλειστικά βάσει κίνησης bundle adjustment.* Υποθέτοντας ότι ισχύουν όλες οι παραδοχές του προηγούμενου υποκεφαλαίου, και αν $e_{i,j}$ είναι το σφάλμα προβολής του j-οστού σημειού του χάρτη στην i-οστή εικόνα, και ω είναι το διάνυσμα που περιέχει την μεταβλητή που θα βελτιστοποιηθεί, δηλαδή την πόζα της κάμερας στην i-οστή εικόνα (i.e. $\{R_{iw}, t_{iw}\}$), τότε για τον αλγόριθμο με σημεία και γραμμές το πρόβλημα βελτιστοποίησης ορίζεται ως εξής:

$$\omega^* = \underset{\omega}{\operatorname{argmin}}\{\sum_{j\in\mathbb{P}} \rho(e_{i,j}^T \Sigma_{e_{i,j}}^{-1} e_{i,j}) + \sum_{k\in\mathbb{L}} \rho(F(\alpha(\mathbb{P}))e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k})\}, \tag{8}$$

όπου $\mathbb{P}$ και $\mathbb{L}$ είναι οι ομάδες που περιέχουν τα αντιστοιχισμένα σημεία και γραμμές, $\rho(.)$ είναι η robust Huber συνάρτηση κόστους, $\alpha(\mathbb{P})$ ο συνολικός αριθμός στοιχείων στην ομάδα $\mathbb{P}$, div είναι η αριθμητική πράξη που δίνει το ακέραιο μέρος μιας διαίρεσης, και $F(\alpha(\mathbb{P}))$ είναι ο προσαρμοστικός παράγοντας ο οποίος διατυπώνεται ως:

$$F(\alpha(\mathbb{P})) = 2^{-\alpha(\mathbb{P})\mathrm{div}50} \tag{9}$$

Η ρύθμιση του προσαρμοστικού παράγοντα έγινε πειραματικά και παρουσιάζεται αναλυτικά στο επόμενο υποκφάλαιο. Στην περίπτωση του αλγορίθμου μόνο με γραμμές, όμοια, το προβλημα βελτιστοποίησης αποτυπώνεται ως εξής:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{k\in\mathbb{L}} \rho(e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k}). \tag{10}$$

2. *Τοπικό bundle adjustment.* Στο τοπικό bundle adjustment, το διάνυσμα ω περιέχει την πόζα της κάμερας για μία ομάδα συνδεδεμένων εικόνων-κλειδιά, όπως και τη θέση των σημείων και γραμμών που παρατηρούνται σε αυτές. Αν το $\mathbb{K}_\mathbb{L}$ περιλαμβάνει την προαναφερθείσα ομάδα εικόνων-κλειδιά, και επίσης τις εικόνες-κλειδιά οι οποίες παρατηρούν κοινά χαρακτηριστικά με τις εικόνες αυτές ορίζονται ως $\mathbb{P}_\mathbb{L}$ και $\mathbb{L}_\mathbb{L}$, αντίστοιχα, τότε η εξίσωση βελτιστοποίησης για τον αλγόριθμο με σημεία και γραμμές είναι η ακόλουθη:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i \in \mathbb{K}_\mathbb{L}} [\sum_{j \in \mathbb{P}_\mathbb{L}} \rho(E_{i,j}) + \sum_{k \in \mathbb{L}_\mathbb{L}} \rho(E_{i,k})], \tag{11}$$

$$E_{i,j} = e_{i,j}^T \Sigma_{e_{i,j}}^{-1} e_{i,j}, \tag{12}$$

$$E_{i,k} = F(\alpha(\mathbb{P}_\mathbb{L})) e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k}. \tag{13}$$

Ενώ για τη μέθοδο μόνο με γραμμές ορίζεται αντίστοιχα:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i \in \mathbb{K}_\mathbb{L}} [\sum_{k \in \mathbb{L}_\mathbb{L}} \rho(e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k})]. \tag{14}$$

4.3 Ρύθμιση του Προσαρμοστικού Παράγοντα

Για τη ρύθμιση του προσαρμοστικού παράγοντα, αξιολογούμε την απόδοση του συστήματος για διάφορες τιμές για το κατώφλι (T) και το βάρος (W), μέσω των οποίων μορφοποιείται ο προσαρμοστικός παράγοντας από τον τύπο: $F(\alpha(\mathbb{P})) = W^{-\alpha(\mathbb{P}) div T}$. Για την αξιολόγηση, επιλέξαμε δύο ακολουθίες εικόνων από το EuRoC dataset [73] και δοκιμάσαμε 6 τιμές για την κάθε παράμετρο. Η πρώτη μας επιλογή ήταν το MH04, το οποίο αποτελεί μία ακολουθία που κατά κύριο λόγο περιλαμβάνει κανονικής υφής, αλλά και μερικά σκοτεινά και χαμηλής υφής σκηνικά, ενώ η δεύτερη ήταν το V203 το οποίο είναι εξαιρετικά απαιτητικό καθώς εμπεριέχει αρκετές θολές εικόνες λόγο απότομων κινήσεων της κάμερας, στις οποίες προφανώς εντοπίζεται χαμηλός αριθμός αντιστοιχιών σημείων. Η ποικιλομορφία στις επιλογές μας απορρέει από το γεγονός ότι θέλαμε να συμπεριλάβουμε όλες τις περιπτώσεις για τη ρύθμιση του προσαρμοστικού παράγοντα. Για τα πειράματα, τρέχουμε κάθε ακολουθία 10 φορές, και τα αποτελέσματα που παρουσιάζονται στον πίνακα 2 αποτελούν τον μέσο όρο των 20 εκτελέσεων.

Κατώφλι \ Παράγοντας Βάρους	1.25	1.5	1.75	2	2.25	2.5
30	0.169	0.158	0.159	0.142	0.158	0.168
40	0.184	0.145	0.139	0.162	0.158	0.183
50	0.213	0.164	0.184	0.131	0.137	0.162
60	0.188	0.172	0.159	0.150	0.175	0.167
70	0.204	0.166	0.150	0.166	0.137	0.147
80	0.257	0.159	0.170	0.176	0.138	0.165

Πίνακας 2: Απόδοση του συστήματος για διαφορετικές τιμές των δύο παραμέτρων του προσαρμοστικού παράγοντα (RMSE ATE in m).

Το σκεπτικό πίσω από τη ρύθμιση του προσαρμοστικού παράγοντα ήταν να ξεκινήσουμε πρώτα με τιμή για το κατώφλι (T) ίση με 30, η οποία είναι η ελάχιστη απαραίτητη τιμή ORB αντιστοιχιών, όπως έχει θεσπιστεί από τον ORB-SLAM3 [1], για να θεωρηθεί έγκυρη η βελτιστοποίηση που αφορά την παρακολούθηση του τοπικού χάρτη. Αναφορικά με το βάρος (W), του δώσαμε τιμές έτσι ώστε το σύστημα να αποδίδει καλύτερα για τις δεδομένες τιμές κατωφλίου. Από τα πειράματα παρατηρήθηκε ότι οι επικρατέστερες τιμές παραμέτρων ήταν 2 για το βάρος, και 50 για το κατώφλι, παρότι το σύστημα λειτουργεί παρόμοια για πολλές διαφορετικές τιμές των παραμέτρων και επομένως είναι ανεκτικό στη μεταβολή τους. Επίσης από τα αποτελέσματα παρατηρήθηκε και μία δεδομένη συσχέτιση μεταξύ των παραμέτρων και της απόδοσης του συστήματος. Συγκεκριμένα, για σταθερό κατώφλι, όσο αυξάνεται το βάρος μειώνεται το σφάλμα, φτάνει σε μία κατώτατη τιμή, και στη συνέχεια ανεβαίνει. Μάλιστα, όπως είναι λογικό για μικρότερες τιμές κατωφλίου συνηθίζεται το σφάλμα να συγκλίνει στην κατώτατη τιμή για μικρότερες τιμές βάρους, και αντίστροφα.

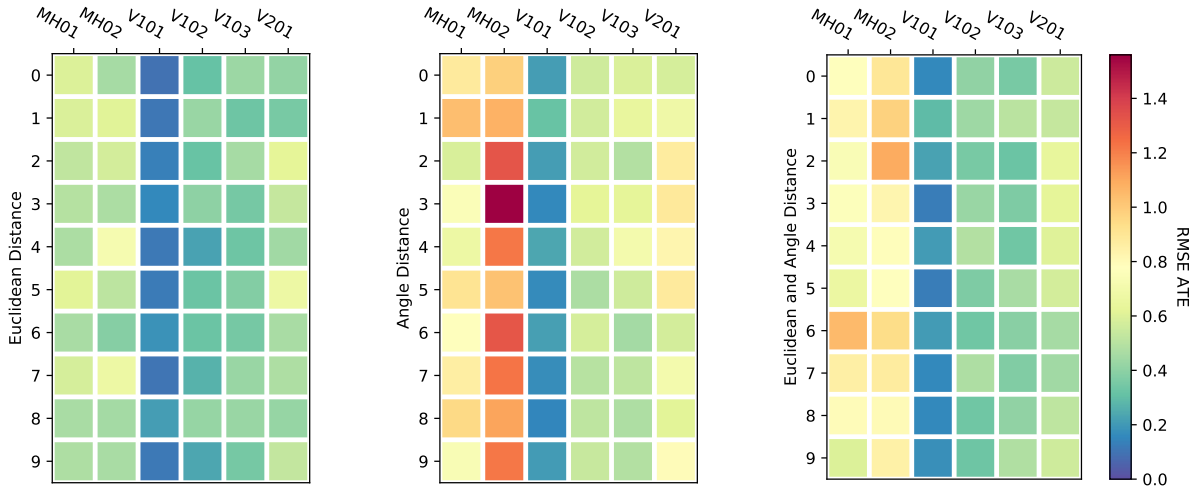
5. Αξιολόγηση

5.1 Αξιολόγηση του Αλγορίθμου Μόνο με Γραμμές

Για την αξιολόγηση της μεθόδου που χρησιμοποιεί μόνο γραμμές, συγκρίνουμε την απόδοσή της σε μερικές ακολουθίες εικόνων του EuRoC dataset [73] για τις τρεις συναρτήσεις σφάλματος που περιγράψαμε στο προηγούμενο κεφάλαιο. Τα αποτελέσματα παρουσιάζονται στην εικόνα 6 και στον πίνακα 3.

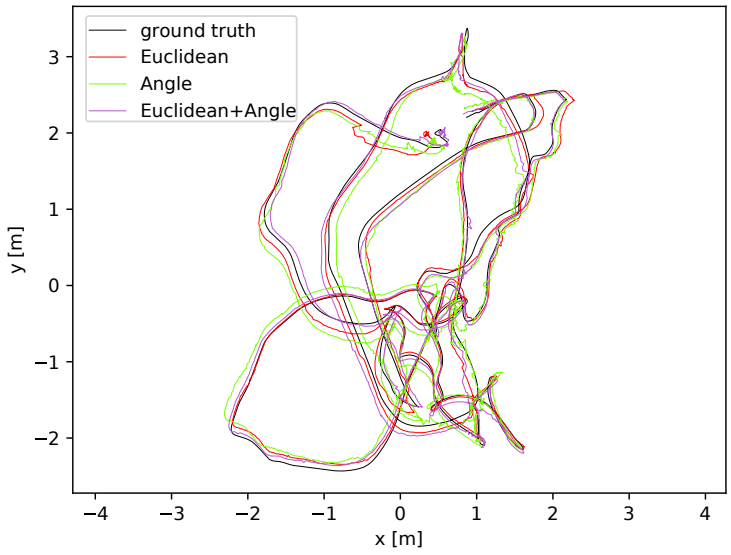
Πίνακας 3: Σύγκριση απόδοσης των διαφορετικών συναρτήσεων σφάλματος στην μέθοδο μόνο με γραμμές. Οι εκτελέσεις έγιναν σε ακολουθίες του EuRoC dataset (RMSE ATE σε m).

Ακολουθία	Ευκλείδεια	Γωνία	Ευκλείδεια και Γωνία
MH01 easy	**0.529**	0.817	0.782
MH02 easy	**0.534**	1.207	0.881
V101 easy	**0.138**	0.208	0.187
V102 medium	**0.328**	0.552	0.398
V103 difficult	**0.386**	0.558	0.403
V201 easy	**0.496**	0.744	0.554
Συνολικός Μέσος Όρος	**0.402**	0.681	0.534



Εικόνα 6: Θερμικοί χάρτες που οπτικοποιούν τα αποτελέσματα των εκτελέσεων για την μέθοδο μόνο με γραμμές. Ο αριστερός χάρτης αντιστοιχεί στην Ευκλείδεια απόσταση, ο μεσαίος στην απόσταση γωνίας, και ο δεξής στον συνδυασμό των δύο. Κάθε χρωματισμένο τετράγωνο αναπαριστά μία εκτέλεση.

Παρουσιάζουμε αποτελέσματα για 6 ακολουθίες του EuRoC, καθώς σχετικά με τις υπόλοιπες, η απόδοση του αλγορίθμου ήταν αρκετά χαμηλή (RMSE ATE μεγαλύτερο του 1.5m) για όλες τις συναρτήσεις σφάλματος, ή δεν κατάφερε να τις ολοκληρώσει, όπως την V203 ακολουθία. Από τα αποτελέσματα είναι προφανές ότι η πιο αποδοτική συνάρτηση σφάλματος είναι αυτή που χρησιμοποιεί αποκλειστικά την Ευκλείδεια απόσταση. Ακόμη, παρατηρείται ότι η συνάρτηση που σχετίζεται με την απόσταση γωνίας δίνει αρκετά μεγαλύτερα σφάλματα και επομένως συνδέεται με μεγαλύτερο drift στον υπολογισμό της πόζας. Τέλος, όπως ήταν λογικό από τα παραπάνω συμπεράσματα, ο συνδυασμός των συναρτήσεων σφάλματος δίνει καλύτερα αποτελέσματα από αυτήν που χρησιμοποιεί την απόσταση γωνίας αλλά χειρότερα από την αντίστοιχη με την Ευκλείδεια απόσταση, γεγονός το οποίο οφείλεται στην εισαγωγή της απόστασης γωνίας η οποία νοθεύει την ποιότητα ακρίβειας στον υπολογισμό της πόζας. Το μόνο πλεονέκτημα του συνδυασμού των δύο, είναι ότι κατά τη βελτιστοποίηση το σύστημα έχει πληροφορία και για την απόσταση γωνίας και επομένως μπορεί να φιλτράρει λίγο πιο αποδοτικά έκτοπες τιμές.



Εικόνα 7: Εκτιμώμενη τροχιά για τις τρεις διαφορετικές συναρτήσεις σφάλματος στην ακολουθία V101. Στην εικόνα απεικονίζεται η πέμπτη καλύτερη εκτέλεση αναφορικά με την κάθε συνάρτηση.

Γενικά, τα μεγαλύτερα σφάλματα παρατηρούμε ότι εντοπίζονται στις ακολουθίες με τη μεγαλύτερη χρονική διάρκεια, γεγονός που οφείλεται στο ότι οι γραμμές λόγω του ότι τα ακραία τους σημεία είναι ασταθή, όπως εξηγήσαμε στο προηγούμενο κεφάλαιο, συνδέονται με μεγαλύτερο σφάλμα σε σύγκριση με τα σημεία. Από την άλλη μεριά, μέχρι και σε δύσκολες ακολουθίες που κινείται απότομα η κάμερα, καταφέρνουν και επιβιώνουν,

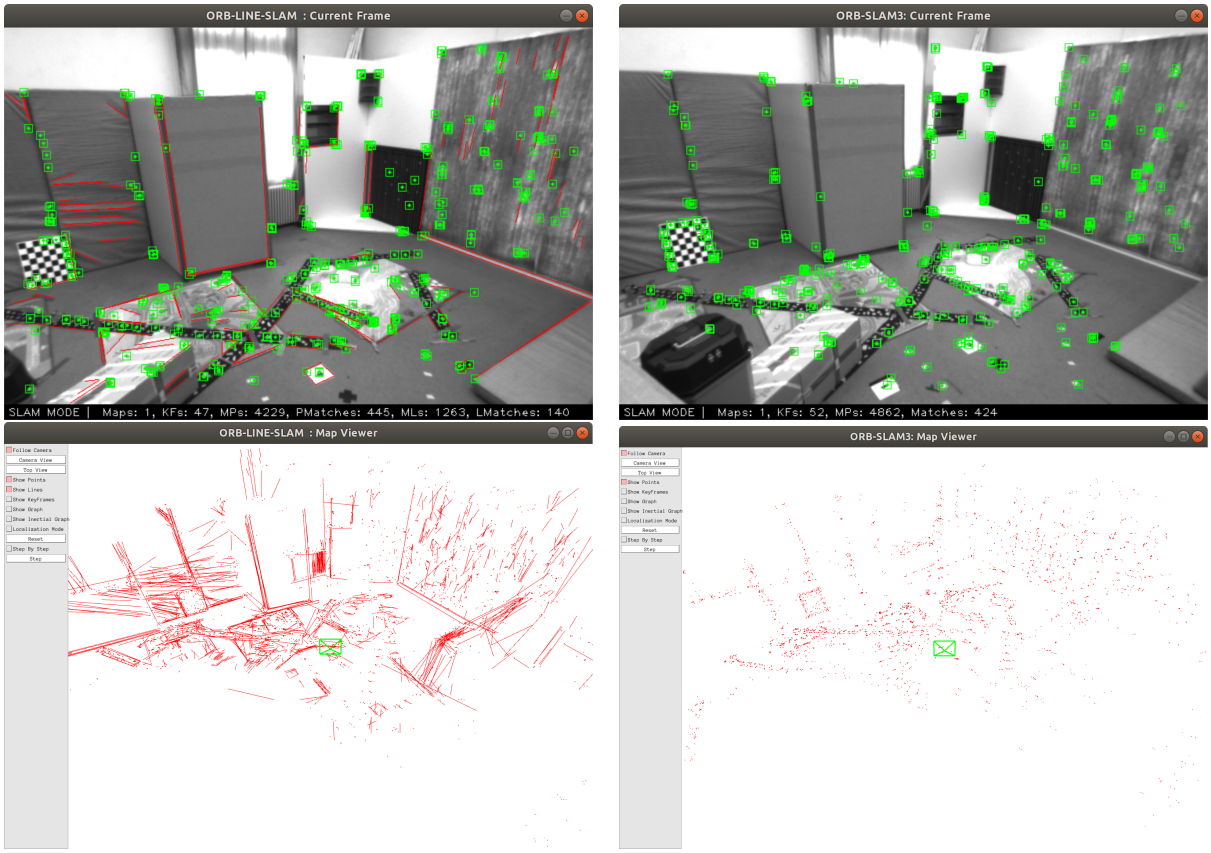
και μάλιστα πετυχαίνουν αξιοσημείωτα αποτελέσματα αν σκεφτεί κανείς ότι δεν κλείνουν βρόχους, όπως κάνει πολλαπλές φορές ο ORB-SLAM3 (παραδείγματος χάριν στο V103).

5.2 Αξιολόγηση του ORB-LINE-SLAM στο EuRoC Dataset

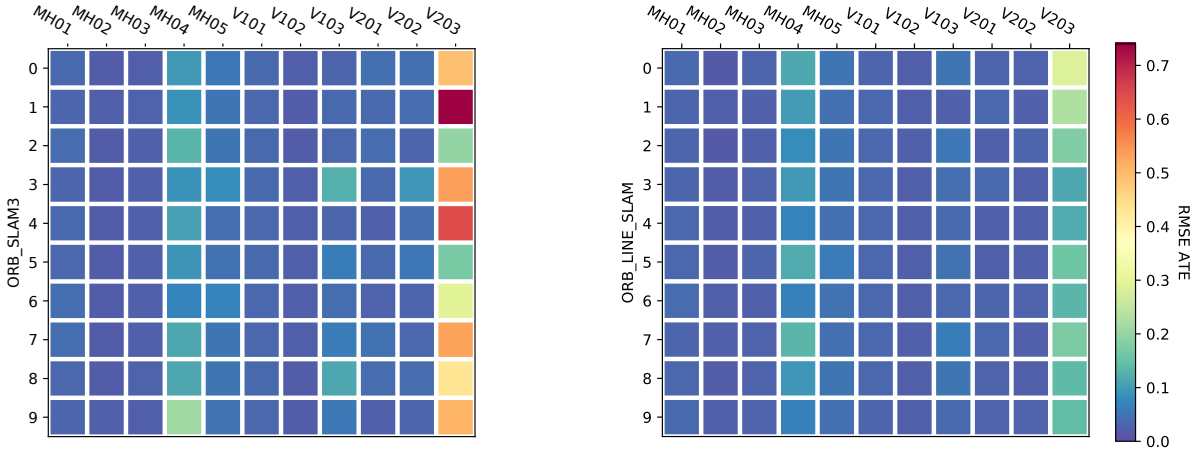
Για την αξιολόγηση του ORB-LINE-SLAM, αρχικά συγκρίνουμε την απόδοση του με αυτήν του ORB-SLAM3 [1] και του PL-SLAM [3] και στις 11 ακολουθίες του EuRoC. Ως μετρικές για την εκτίμηση των αλγορίθμων χρησιμοποιούμε το root mean squared absolute trajectory error (RMSE ATE), το οποίο εκφράζει σφάλμα μεταξύ της εκτιμώμενης και της πραγματικής τροχιάς, και το root mean squared relative translational error, το οποίο εκφράζει το drift του υπολογισμού της πόζας από τη μία εικόνα στην επόμενη. Για τον PL-SLAM χρησιμοποιούμε τα ήδη υπάρχοντα αποτελέσματα καθώς ο open-source αλγόριθμος είναι ημιτελής και επομένως δεν μπορέσαμε να τον τρέξουμε. Επίσης παραθέτουμε και τη συνολική κάλυψη σχετικά με την ακολουθία V203, η οποία εκφράζεται ως το κλάσμα των εικόνων που κατάφερε ο αλγόριθμος να προσδιορίσει την πόζα της κάμερας προς το συνολικό αριθμό εικόνων που θα μπορούσε να την έχει προσδιορίσει. Στις υπόλοιπες ακολουθίες οι αλγόριθμοι πέτυχαν 100% κάλυψη. Εκτός από την ποσοτική αξιολόγηση, είναι άξιο αναφοράς ότι ποιοτικά η υλοποίηση μας χτίζει πολύ πιο ρεαλιστικές αναπαραστάσεις του περιβάλλοντος σχετικά με τον ORB-SLAM3, όπως φαίνεται και στην εικόνα 8. Τα αποτελέσματα της απόδοσης των αλγορίθμων παρουσιάζονται στον πίνακα 4 και στην εικόνα 9.

Πίνακας 4: Σύγκριση απόδοσης στο EuRoC dataset. Το t_{rel} εκφράζει το RMSE relative translational error σε m, και το t_{abs} το RMSE ATE σε m.

Ακολουθία	ORB-LINE-SLAM		ORB-SLAM3		PL-SLAM
	t_{rel}	t_{abs}	t_{rel}	t_{abs}	t_{rel}
MH01 easy	**0.038**	**0.033**	0.039	0.035	0.042
MH02 easy	**0.043**	0.019	**0.043**	0.019	0.052
MH03 medium	0.074	0.027	0.075	**0.025**	**0.040**
MH04 difficult	0.068	**0.094**	0.068	0.111	**0.064**
MH05 difficult	**0.064**	**0.049**	**0.064**	0.054	0.070
V101 easy	**0.023**	**0.033**	**0.023**	0.035	0.042
V102 medium	0.054	0.022	0.056	**0.021**	**0.046**
V103 difficult	**0.046**	**0.044**	0.048	0.059	0.069
V201 easy	**0.025**	**0.029**	**0.025**	0.036	0.061
V202 medium	**0.051**	**0.024**	0.052	0.044	0.056
V203 difficult	**0.066**	**0.167**	0.076	0.453	0.126
Κάλυψη	**92%**		88%		
Συνολικός Μέσος Όρος	**0.050**	**0.049**	0.052	0.081	0.061



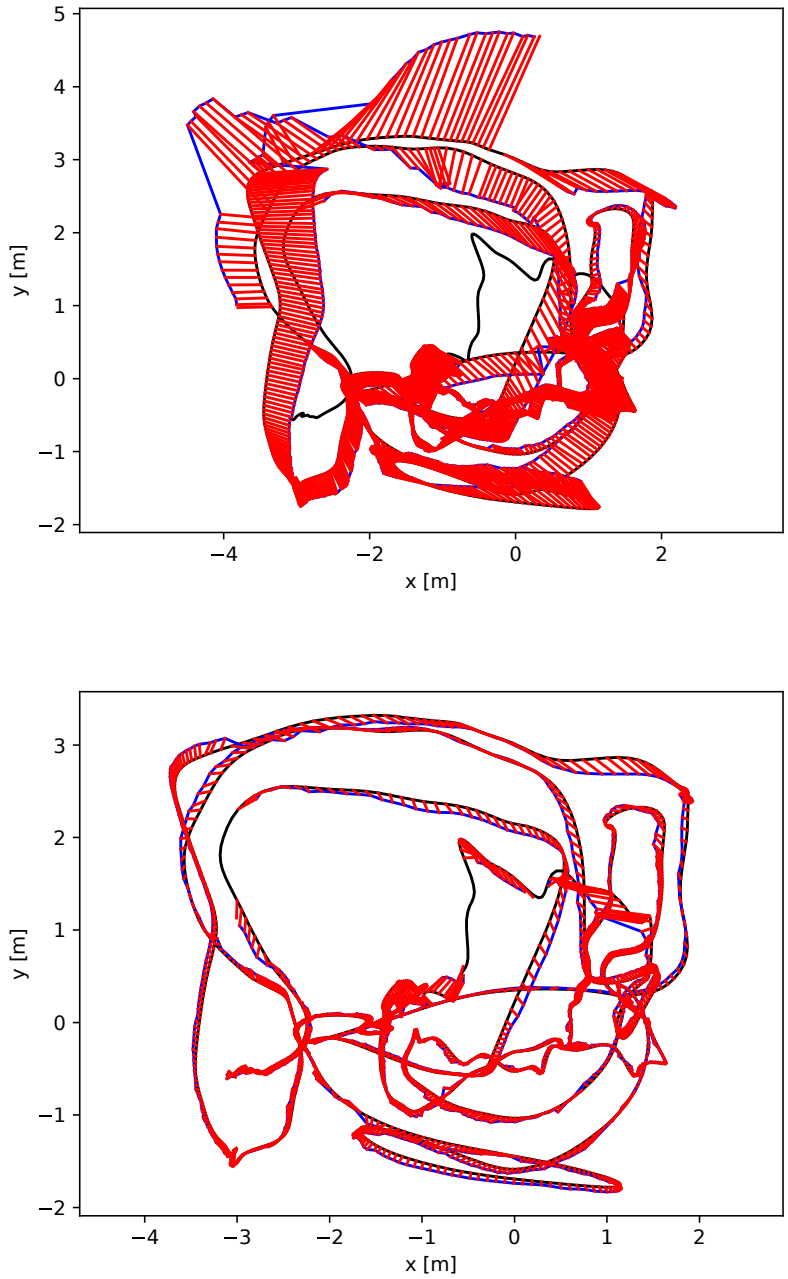
Εικόνα 8: Ανακατασκευή του ίδιου σκηνικού από τον ORB-LINE-SLAM (αριστερά) και τον ORB-SLAM3 (δεξιά).



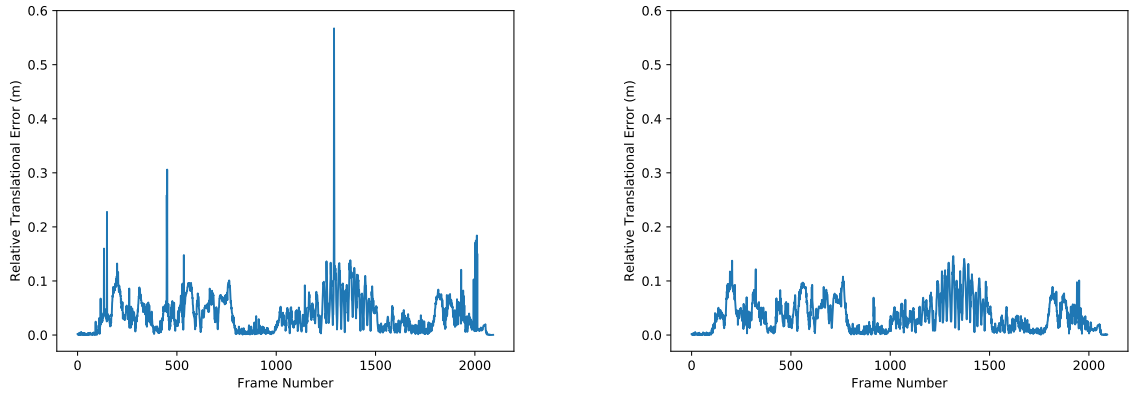
Εικόνα 9: Θερμικοί χάρτες που αναπαριστούν όλες τις εκτελέσεις για το EuRoC dataset. Ο αριστερός χάρτης αντιστοιχεί στον ORB-SLAM3 ενώ ο δεξής στον ORB-LINE-SLAM. Κάθε χρωματισμένο τετράγωνο αναπαριστά μία εκτέλεση.

Όπως φαίνεται από τα παραπάνω αποτελέσματα, ο ORB-LINE-SLAM κυριαρχεί εμφανώς έναντι του ORB-SLAM3, και η μεγαλύτερη διαφορά μεταξύ των δύο φαίνεται στην ακολουθία V203 η οποία είναι αρκετά απαιτητική καθώς περιέχει αρκετά θολά σκηνικά λόγω απότομων κινήσεων της κάμερας, αλλά και λείπουν κάποιες εικόνες από τη μία κάμερα. Συγκεκριμένα, ο αλγόριθμός μας υπερτερεί τόσο σε όρους RMSE ATE και RMSE relative translational error, όσο και στην μέση κάλυψη. Το γεγονός αυτό δηλώνει πόσο δραστικό ρόλο παίζουν οι γραμμές όταν η μέθοδος δεν προσδιορίζει αρκετές αντιστοιχίες σημείων, αλλά και επίσης παρατηρείται ότι σε περιπτώσεις που έχει χαθεί η παρακολούθηση της πόζας της κάμερας, όπως για παράδειγμα στην τελευταία στροφή στη V203, ο αλγόριθμος με τις γραμμές ανακτά γρηγορότερα την εκτίμηση της πόζας με αποτέλεσμα να είναι ικανός να εντοπίζει βρόχο πολύ συχνότερα σε σύγκριση με τον ORB-SLAM3. Τα παραπάνω συμπερασματικά σχόλια είναι εμφανή στην εικόνα στην εικόνα 10. Σύμφωνα με τον πίνακα, ο αλγόριθμός μας είναι ικανότερος ικανότερος από τον ORB-SLAM3 συγκρίνοντας με βάση και τις δύο επιλεγμένες μετρικές. Αναφορικά με το σχετικό σφάλμα, ενώ ο ORB-LINE-SLAM υπερτερεί ή είναι τουλάχιστον ίσος με τον ORB-SLAM3 σε όλες τις ακολουθίες, παρατηρούμε σχετικά μικρές διαφορές. Αυτό όμως είναι λογικό αν σκεφτεί κανείς ότι η μετρική αυτή εκφράζει drift από εικόνα σε εικόνα το οποίο όμως όσο περνάει ο χρόνος ολοκληρώνεται και αποτυπώνεται ως σημαντικό σφάλμα μεταξύ εκτιμώμενης και πραγματικής τροχιάς. Ένα τέτοιο παράδειγμα αποτυπώνεται στην ακολουθία V103, όπου ενώ το σχετικό σφάλμα μεταξύ των αλγορίθμων είναι κοντινό, όπως φαίνεται από την εικόνα 11, οι γραμμές εξομαλύνουν κάποιες αιχμές στο drift οι οποίες οδηγούν σε μεγαλύτερο RMSE ATE για τον ORB-SLAM3. Η μικρή διαφορά στο σχετικό σφάλμα οφείλεται στο ότι οι αιχμές αυτές είναι ελάχιστες συγκριτικά με το συνολικό αριθμό μετρήσεων που συνεισφέρουν στο σφάλμα οι οποίες είναι κατά προσέγγιση κάποιες χιλιάδες.

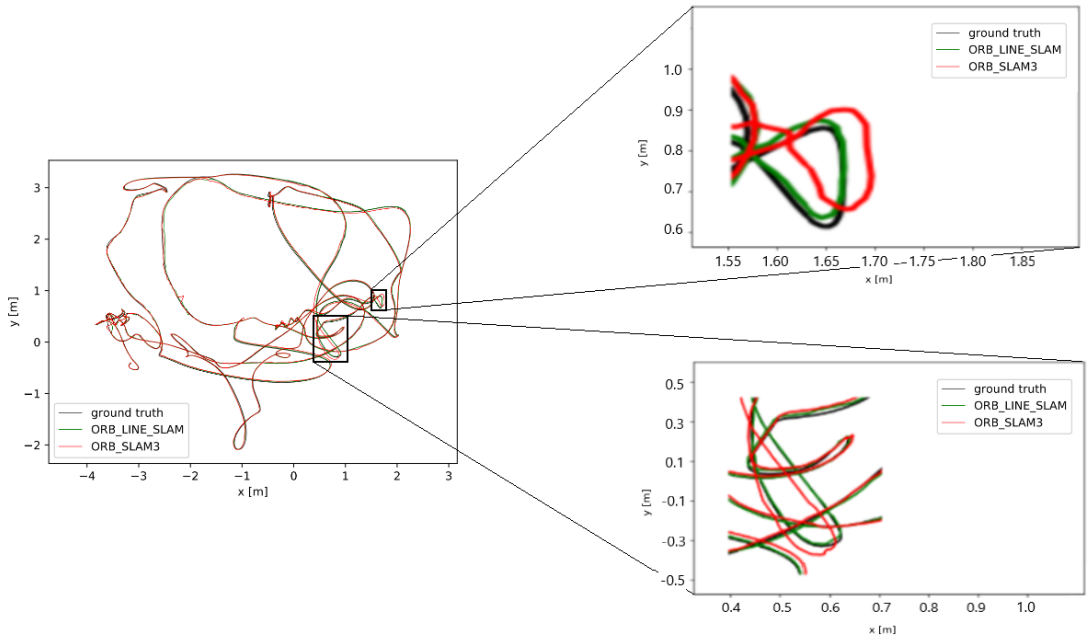
Συγκρίνοντας τα αποτελέσματα του πίνακα, είναι διακριτό ότι ο ORB-LINE-SLAM επικρατεί σημαντικά έναντι του PL-SLAM. Συγκεκριμένα, ο PL-SLAM δίνει καλύτερο σχετικό σφάλμα μόνο σε 3 ακολουθίες, εκ των οποίων στις δύο σε κλίμακα χιλιοστών, και μόνο σε μία (MH03) σε κλίμακα εκατοστών (3.4 εκατοστά). Από την άλλη, η υλοποίησή μας υπερνικά τον PL-SLAM στις υπόλοιπες 8 ακολουθίες, και συγκεκριμένα στις μισές από αυτές, σε κλίμακα εκατοστών (V101 για 1.9 εκατοστά, V103 για 2.3 εκατοστά, V201 για 3.6 εκατοστά). Η μεγαλύτερη διαφορά μεταξύ των δύο παρατηρείται και πάλι στη V203, όπου το σύστημά μας επιτυγχάνει σχετικό σφάλμα 6 εκατοστά μικρότερο. Το συνολικό μέσο σφάλμα του αλγορίθμου μας είναι 1.1 εκατοστά χαμηλότερο από αυτό του PL-SLAM.



Εικόνα 10: Εκτιμώμενη τροχιά (μπλε), πραγματική τροχιά (μαύρη), και η διαφορά της πόζας μεταξύ αντίστοιχων εικόνων (κόκκινη), στη V203. Η πάνω εικόνα αποτυπώνει την πέμπτη καλύτερη εκτέλεση του ORB-SLAM3 και η ακριβώς από κάτω την πέμπτη καλύτερη του ORB-LINE-SLAM.



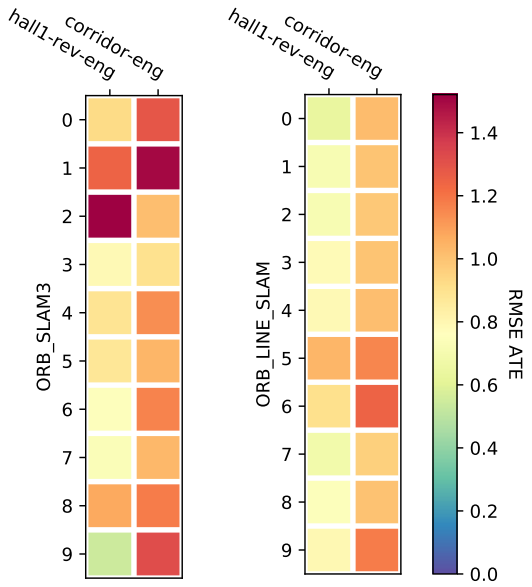
Εικόνα 11: Σχετικό σφάλμα μετατόπισης του ORB-SLAM3 (αριστερά) και του ORB-LINE-SLAM (δεξιά) στη V103. Και οι δύο εικόνες αντιστοιχούν στην πέμπτη καλύτερη εκτέλεση.



Εικόνα 12: Εκτιμώμενη τροχιά του ORB-SLAM3 και του ORB-LINE-SLAM στη V202. Η εικόνα αποτυπώνει την πέμπτη καλύτερη εκτέλεση των αλγορίθμων.

5.3 Αξιολόγηση του ORB-LINE-SLAM στο UMA Dataset

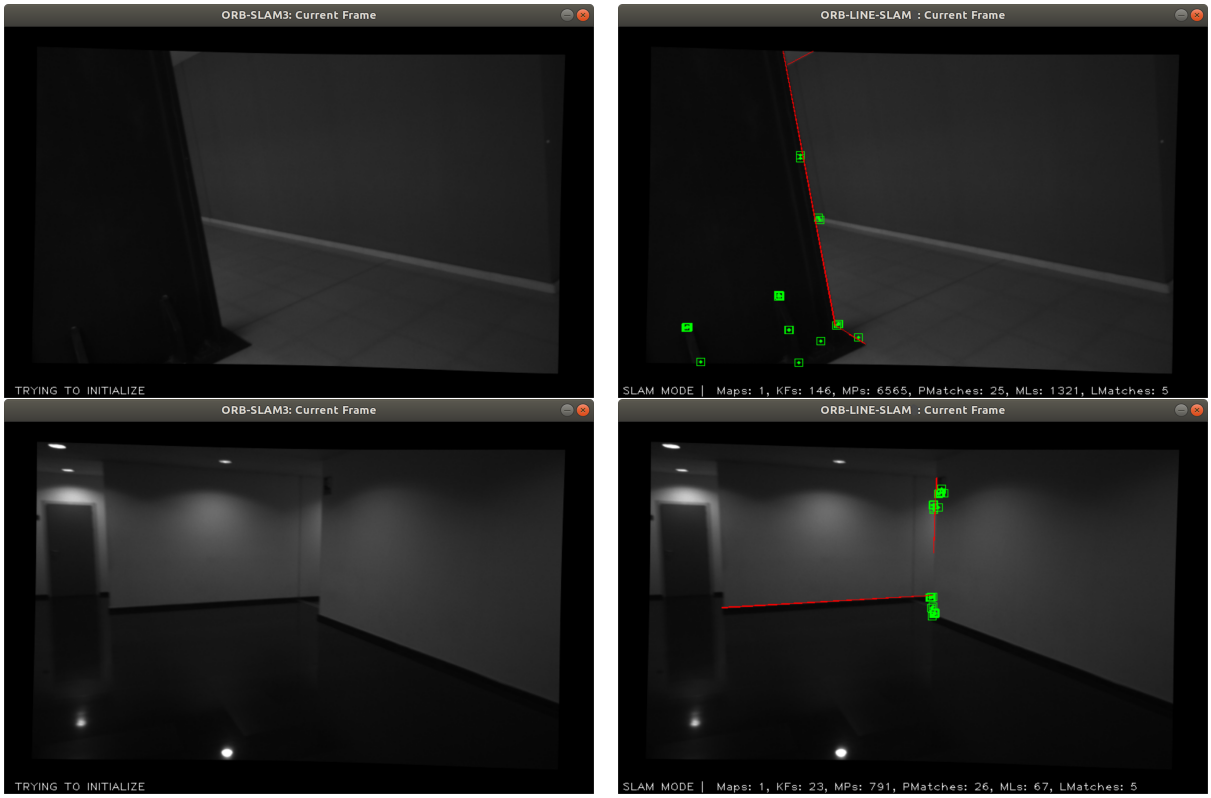
Για την περαιτέρω αξιολόγηση του συστήματός μας επιλέξαμε δύο απαιτητικές ακολουθίες εικόνων του UMA dataset [74]. Το UMA περιέχει ευρεία γκάμα ακολουθιών που απευθύνονται σε δύσκολα σενάρια για το πρόβλημα του SLAM, όπως για παράδειγμα χαμηλής υφής χώρους, διαρκής εναλλαγές φωτισμού, υπερβολική έκθεση στο ηλιακό φως, και υπερβολικά απότομες κινήσεις της κάμερας. Και οι δύο ακολουθίες που διαλέξαμε για την αξιολόγηση περιλαμβάνουν χαμηλής υφής και σκοτεινά σκηνικά τα οποία τις καθιστούν πρόκληση για πολλές SLAM μεθόδους. Λόγω του ότι σε όλες τις ακολουθίες του UMA ο αισθητήρας κάνει περίπου μία κυκλική διαδρομή και στο τέλος επιστρέφει στο αρχικό σημείο, και του γεγονότος ότι το ground-truth περιέχει μόνο εικόνες από την αρχή και το τέλος της ακολουθίας, για την αξιολόγηση των αλγορίθμων απενεργοποιούμε το στάδιο σχετικά με το κλείσιμο βρόχου (loop closing thread), διότι σε αντίθετη περίπτωση το σύστημα πάντα στο τέλος θα έκλεινε βρόχο και επομένως δεν θα μπορούσαμε να αξιολογήσουμε το πραγματικό drift. Ακόμη, υποθέτοντας ότι τα δείγματα είναι ανεξάρτητα και έχουν ίσες διακυμάνσεις, εφαρμόζουμε το t-test [75] στα αποτελέσματα των δύο αλγορίθμων ώστε να προσδιορίσουμε τη στατιστική σημαντικότητα της διαφοράς τους σε όρους απόδοσης κάτω από απαιτητικές συνθήκες. Ο πίνακας 5 και η εικόνα 13 δείχνουν την απόδοση των συστημάτων στις δύο επιλεγμένες ακολουθίες (hall1-rev-eng και corridor-eng) του UMA dataset.



Εικόνα 13: Θερμικοί χάρτες που αναπαριστούν όλες τις εκτελέσεις που έγιναν στο UMA dataset. Ο αριστερός χάρτης αντιστοιχεί στον ORB-SLAM3 και ο δεξής στον ORB-LINE-SLAM. Κάθε χρωματισμένο τετράγωνο αναπαριστά μία εκτέλεση.

Πίνακας 5: Σύγκριση απόδοσης στο UMA dataset (RMSE ATE σε m). Το RMSE ATE αναπαριστά τον μέσο όρο 10 πετυχημένων εκτελέσεων (± την τυπική απόκλιση) και τα ποσοστά επιτυχίας αντιστοιχούν μόνο στις πρώτες 10 εκτελέσεις.

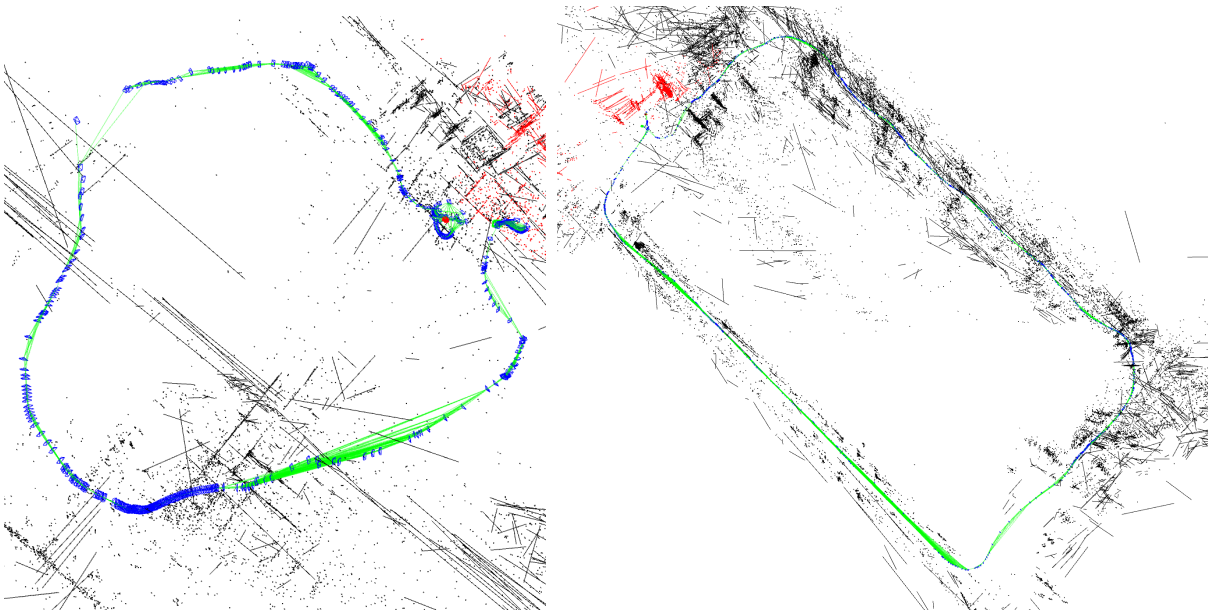
Ακολουθία	ORB-LINE-SLAM	ORB-SLAM3	P του t-test
hall1-rev-eng	**0.780 (±0.116)**	0.934 (±0.162)	0.025
Ποσοστό πετυχημένων εκτελέσεων	**100%**	30%	
corridor-eng	**1.055 (±0.099)**	1.156 (±0.110)	0.044
Ποσοστό πετυχημένων εκτελέσεων	**100%**	90%	



Εικόνα 14: Στιγμιότυπα από τις ακολουθίες *hall1-rev-eng* και *corridor-eng*. Οι πάνω εικόνες αντιστοιχούν στην *hall1-rev-eng* και οι κάτω στην *corridor-eng*. Οι φωτογραφίες στα αριστερά αναφέρονται στον ORB-SLAM3 και αυτές στα δεξιά στον ORB-LINE-SLAM.

Όπως συμπεραίνεται από τα παραπάνω αποτελέσματα, το προτεινόμενο σύστημα είναι πολύ πιο ακριβές και σταθερό σε χαμηλής υφής περιβάλλοντα και σε ακολουθίες που περιλαμβάνουν έντονες αλλαγές στο φωτισμό. Αναλυτικά, λόγω των επιπλέον αντιστοιχιών από τις γραμμές, το σύστημα καταφέρνει να επιβιώσει σε σκοτεινά σκηνικά στα οποία ο ORB-SLAM3 πολλές φορές αποτυγχάνει. Αυτό αποτυπώνεται στην εικόνα 14. Εκτός

αυτών, όπως επιβεβαιώνεται από τις P τιμές των t-tests οι οποίες είναι χαμηλότερες του 5%, το σύστημα μας επιτυγχάνει με συνέπεια χαμηλότερο σφάλμα και η διαφορά μεταξύ των αποτελεσμάτων δεν είναι προϊόν τύχης. Επειδή το ground-truth περιέχει εικόνες μόνο από την αρχή και το τέλος της ακολουθίας δεν μπορέσαμε να δημιουργήσουμε συγκριτικές αναπαραστάσεις τροχιάς για τις δύο μεθόδους. Παρόλα αυτά, οι τροχιές και οι ανακατασκευές του περιβάλλοντος για τις δύο αυτές ακολουθίες χρησιμοποιώντας τον ORB-LINE-SLAM, περιγράφονται στην εικόνα 15.



Εικόνα 15: Τροχιές και ανακατασκευές του περιβάλλοντος αναφορικά με τις ακολουθίες *hall1-rev-eng* (αριστερά) και *corridor-eng* (δεξιά).

5.4 Υπολογιστικός Χρόνος

Ως τελευταίο βήμα για την αξιολόγηση του προτεινόμενου συστήματος, μετρήσαμε τον χρόνο που απαιτείται για να τρέξουν οι δύο ενότητες του συστήματος που περιλαμβάνονται οι γραμμές. Στον πίνακα 6 παρουσιάζονται τα αποτελέσματα υπολογιστικού χρόνου για διαφορετικές ακολουθίες των EuRoC και UMA datasets. Από τους εξαγόμενους χρόνους, συμπεραίνουμε ότι το σύστημα μας είναι ικανό να τρέξει σε πραγματικό χρόνο καθώς ο μέσος χρόνος παρακολούθησης (tracking) κάθε εικόνας είναι μικρότερος από τον αντίστροφο αριθμό της συχνότητας της κάμερας. Ωστόσο, όπως ήταν αναμενόμενο, η προθήκη των γραμμών ανέβασε τον υπολογιστικό χρόνο, κυρίως λόγο της διαδικασίας εντοπισμού τους. Όμως, το εμπόδιο αυτό μπορεί εύκολα να ξεπεραστεί από το χρήστη μειώνοντας την αρχική κλίμακα της εικόνας από την οποία εξάγονται οι γραμμές, με μικρή σχετικά θυσία στην

απόδοση της μεθόδου. Έτσι, το σύστημα μας είναι ικανό να λειτουργήσει και σε λιγότερο αποδοτικές CPU's.

Πίνακας 6: Χρόνος εκτέλεσης των ενοτήτων παρακολούθησης (Tracking) και τοπικής χαρτογράφησης (Local Mapping) του ORB-LINE-SLAM και του ORB-SLAM3 (μέσος όρος ± τυπική απόκλιση σε ms).

Ρυθμίσεις	Ακολουθία	V103	V203	corridor-eng
	Dataset	EuRoC	EuRoC	UMA
	Ανάλυση	752×480	752×480	752×480
	Συχνότητα Κάμερας (Hz)	20	20	25

ORB-LINE-SLAM

Tracking	Εξαγωγή Χαρακτηριστικών	28.88 ± 6.62	29.27 ± 7.41	20.35 ± 5.76
	Stereo Αντιστοίχιση	9.09 ± 2.87	9.30 ± 2.67	8.28 ± 2.58
	Πρόβλεψη Πόζας	3.09 ± 1.20	3.08 ± 1.11	2.75 ± 1.00
	Παρακολούθηση Τοπικού Χάρτη	7.43 ± 3.76	5.86 ± 3.58	6.06 ± 2.92
	Απόφαση Νέας Εικόνας-Κλειδί	0.05 ± 0.05	0.05 ± 0.09	0.04 ± 0.10
	Σύνολο	49.60 ± 11.01	48.93 ± 11.35	38.42 ± 11.02
Local Mapping	Εισαγωγή Εικόνας-Κλειδί	6.16 ± 1.76	5.45 ± 1.92	11.82 ± 6.02
	Αποκοπή Χαρακτηριστικών του Χάρτη	0.35 ± 0.15	0.26 ± 0.10	0.14 ± 0.07
	Δημιουργία Νέων Σημείων	32.75 ± 16.54	24.25 ± 12.86	35.26 ± 15.21
	Τοπικό Bundle Adjustment	95.20 ± 90.41	44.69 ± 46.78	85.96 ± 115.36
	Αποκοπή Εικόνων-Κλειδιά	3.33 ± 2.66	1.93 ± 1.95	3.43 ± 2.80
	Σύνολο	144.29 ± 108.00	82.28 ± 59.25	148.67 ± 128.51

ORB-SLAM3

Tracking	Εξαγωγή Χαρακτηριστικών	11.68 ± 2.80	11.93 ± 3.94	10.43 ± 1.75
	Stereo Αντιστοίχιση	8.41 ± 2.76	8.57 ± 2.93	7.39 ± 2.37
	Πρόβλεψη Πόζας	2.01 ± 0.65	1.98 ± 0.81	1.75 ± 0.61
	Παρακολούθηση Τοπικού Χάρτη	7.03 ± 3.10	5.13 ± 2.78	4.61 ± 2.21
	Απόφαση Νέας Εικόνας-Κλειδί	0.04 ± 0.08	0.04 ± 0.09	0.03 ± 0.08
	Σύνολο	30.20 ± 5.51	28.70 ± 6.97	25.03 ± 5.55
Local Mapping	Εισαγωγή Εικόνας-Κλειδί	5.91 ± 1.56	5.29 ± 2.00	10.05 ± 4.55
	Αποκοπή Χαρακτηριστικών του Χάρτη	0.25 ± 0.10	0.23 ± 0.09	0.10 ± 0.04
	Δημιουργία Νέων Σημείων	33.21 ± 15.48	21.67 ± 14.34	31.54 ± 13.08
	Τοπικό Bundle Adjustment	80.92 ± 77.15	40.34 ± 56.54	54.74 ± 63.70
	Αποκοπή Εικόνων-Κλειδιά	4.34 ± 3.46	1.91 ± 2.78	2.38 ± 2.35
	Σύνολο	130.79 ± 95.62	74.95 ± 72.49	108.95 ± 75.88

6. Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα

Στην παρούσα διπλωματική, προτείναμε τον ORB-LINE-SLAM, ένα υβριδικού τύπου σύστημα οπτικού SLAM σύστημα που έχει τη δυνατότητα να λειτουργήσει α) αποκλειστικά με γραμμές ή β) συνδυάζοντας ταυτόχρονα με έναν προσαρμοστικό τρόπο σημεία και γραμμές. Επίσης, παρουσιάσαμε έναν νέο πειραματικά ρυθμισμένο προσαρμοστικό παράγοντα με σκοπό τον πιο αποδοτικό συνδυασμό σημείων και γραμμών. Εκτός αυτού, συγκρίναμε σε βάθος διαφορετικούς αλγορίθμους για τον εντοπισμό των γραμμών όπως και διαφορετικές συναρτήσεις σφάλματος για το bundle adjustment, και εξήγαμε αποτελέσματα και συμπεράσματα σχετικά με την απόδοσή τους αναφορικά με το πρόβλημα του οπτικού SLAM. Τα πειράματά μας υποδεικνύουν ότι η μέθοδός μας που χρησιμοποιεί σημεία και γραμμές ξεπερνάει σημαντικά σε όρους απόδοσης τον PL-SLAM [3] και βελτιώνει την ακρίβεια του ORB-SLAM3 [1], ειδικά σε απαιτητικές ακολουθίες εικόνων που περιέχουν χαμηλής υφής σκηνικά ή εικόνες που το σκηνικό είναι υπέρμετρα θολό λόγω απότομης κίνησης της κάμερας. Τέλος, η μέθοδός μας που χρησιμοποιεί μόνο γραμμές, από όσο γνωρίζουμε, αποτελεί το πρώτο open-source σύστημα οπτικού SLAM αποκλειστικά με γραμμές που είναι ικανό να λειτουργήσει με ικανοποιητική ακρίβεια.

6.2 Μελλοντικές Επεκτάσεις

Μελλοντική έρευνα πάνω στην εργασία αυτή, θα μπορούσε να εστιάσει στην αντικατάσταση του συμβατικού LBD [69] descriptor γραμμών, με νεότερες και πιο αποδοτικές λύσεις που βασίζονται στη μηχανική μάθηση, όπως για παράδειγμα οι DLD[76] και WLD[77], και οι οποίες θα μπορούσαν να βελτιώσουν σημαντικά τη διαδικασία της αντιστοίχισης γραμμών. Επιπρόσθετα, η εισαγωγή ενός πιο στοχευμένου στο πρόβλημα του οπτικού SLAM detector γραμμών, όπως για παράδειγμα αυτός που παρουσιάζεται στο [78], όπου θα δίνει βάση στον εντοπισμό των ίδιων ευθύγραμμων τμημάτων με όσο γίνεται πιο σταθερά ακραία σημεία, θα βοηθούσε εξαιρετικά στην απόδοση του συστήματος και στη μείωση του υπολογιστικού κόστους, καθώς οι αλγόριθμοι detectors που χρησιμοποιούνται αυτή τη στιγμή εστιάζουν στην ρεαλιστική απεικόνιση του περιβάλλοντος και όχι στο πρόβλημα του SLAM. Πέρα από τα παραπάνω, η υλοποίησή μας θα μπορούσε να επεκταθεί, τόσο για διαφορετικού τύπου κάμερες (Monocular, Fisheye, RGB-D), όσο και για να συμπεριλάβει διαφορετικούς αισθητήρες, όπως για παράδειγμα IMU ή LiDAR, με σκοπό να βελτιωθεί η ανεκτικότητα του συστήματος κάτω από εξαιρετικά απαιτητικές συνθήκες.

Chapter 1

Introduction

1.1 Brief Preview of History of SLAM

Simultaneous Localization and Mapping (SLAM) comprises the concurrent estimation of the state of a robot (which can include pose, velocity, sensor biases, and calibration parameters) equipped with on-board sensors and reconstruction of a map of its surrounding environment that the sensors are perceiving. Concerning modern visual SLAM, in most cases, the state of the robot is described by its pose (position and orientation), while the only sensor that is used is a camera. The map represents the aspects of interest (e.g., position of landmarks, obstacles) that define the surrounding environment that the robot constructs.

The usefulness of a map for a SLAM system is vital. First and foremost, the map is crucial for the minimization of the accumulated drift that is committed during the estimation of the pose of the robot. For example, with the use of a map that contains a set of distinguishable landmarks, the robot by re-visiting known areas in which common landmarks are visible, can detect and close loops, eliminating this way the accumulated error by optimizing the pose of the camera throughout the whole loop. Second, the map can give supplementary information about the texture of the environment, such as the existence of obstacles, and inform the path planning or even provide an intuitive visualization for a human operator.

The popularity that SLAM gained during the last decades is not unexpected if one thinks about the manifold aspects that it involves. At the lower level, SLAM intersects other research fields such as computer vision, signal processing, and artificial intelligence. At the higher level it combines assorted topics such as geometry, graph theory, optimization, and probabilistic estimation. Furthermore, SLAM is also connected with practical issues ranging from sensor calibration to system integration. This variability of aspects and scientific fields that SLAM is connected to, in combination with the tremendous variability of possible environments and the factor of uncertainty that is always infused to them, indisputably categorizes SLAM to one of the biggest modern challenges in the field of robotics. As a result, it is logical that the initial conducted research approached static environments and focused mainly on probabilistic methods for the localization of the robot. Subsequently, extensive research and the availability of large memory spaces, has made it possible to store a large number of observations gathered by the robot and adopt a nonlinear optimization framework in order to give a solution to the SLAM problem. Modern SLAM can be classified into two distinct categories: LiDAR SLAM, and visual SLAM. LiDAR SLAM is based on a light detection and ranging sensor and is considered the most reliable and precise option in the field. Its main advantages yield to the ability of LiDAR sensor to build point clouds with accurate range perception and robustness against lighting conditions. However, its demands on computational resources in conjunction with the excessive cost of LiDAR, render LiDAR SLAM prohibitive for many applications. On the other hand, visual SLAM by using a camera as a sensor, although it is sensitive to lighting conditions, it provides a stable solution by taking advantage of

the rich visual information and requiring less computational power while it is much more cost-efficient. For this reason, visual SLAM constitutes the typical choice in the field, while it is associated with a wide range of applications. In visual SLAM, the position of the camera is estimated by minimizing either the photometric error between corresponding sets of pixels of consecutive frames (direct methods), or the re-projection error between corresponding features (feature-based methods). Over the last years, the most dominant algorithm seems to be the later which is thoroughly researched and extended in numerous ways. Particularly, an increasing research interest has been focusing on developing algorithms that combine visual geometric features (e.g., line segments) with points, which usually constitute the typical choice of preference in the field. The main benefits of this approach are the ability of the system to survive in low-textured environments, which although poor in point features are rich in other geometric features, and the reconstruction of physically more meaningful representation of the environment. Even more recent approaches, through machine learning methods incorporate into the feature-based SLAM algorithms semantic information about the scene by classifying objects with labels as landmarks. Specifically, they bolster the ability of mobile robots to interpret the environment (an attribute that would be very useful for autonomous vehicles which have to make decisions according to how they perceive their surrounding space) and additionally produce enhanced maps by abstracting objects with their geometric shapes. Finally, visual-inertial algorithms have caught swiftly the attention of the industry as with the addition of just an inertial measurement unit (IMU) which provides the system with self-motion information, visual methods manage to survive even in challenging environments where the visual information is not sufficient. A more detailed overview of different periods of the history of SLAM can be found in [79, 80, 81, 82].

1.2 Challenges in Modern Visual SLAM

Simultaneous Localization and Mapping is one of the most fundamental capabilities necessary for robots as various modern application are inextricably connected with it. Due to the ubiquitous availability of images, visual SLAM has become a principal component for the autonomy of many robotic systems. For applications ranging from virtual and augmented reality to fully autonomous vehicles, visual SLAM constitutes one of the most efficient and low-cost options in the field of robotics. Many modern challenges including exploring environments in which human access is impossible, such as space exploration or pit and ramshackle mine inspection or even human rescue in impassable areas, transportation of disabled people through autonomous vehicles, and low-cost and quick inspection for maintenance, namely examination of damaged industrial areas or eroded cargo holds and ballast tanks of ships, commence to be solved through visual SLAM.

Nevertheless, real-life environments are full of difficult cases such as changing or lack

of illumination, dynamic objects, and texture-less scenes (Figure 1.1). Under such circumstances visual SLAM algorithms fail to provide stable and accurate results while their robustness is significantly poorer. The main obstacles yield to different causes for each occasion. Concerning illumination changes, the gradual alterations in the density of the pixels notably decrease the efficacy of feature-based methods as the system fails to repeatedly detect and match same features, while in the case of direct methods whose foundation is the hypothesis of invariable gray scale, the change in gray-scale results in remarkably poorer robustness. As for the texture-less scenes, direct algorithms due to the fact that they are not so dependent to the geometrical structures of the scene, they are more precise and robust in comparison to traditional feature-based methods which exclusively depend on the ability of the system to detect and match point features even if they are scanty in the scene. Lastly, of course it is apparent that lack of illumination means insufficient visual information and thus all visual SLAM algorithms fail in such conditions. The community has been working on many solutions to these issues mostly by trying to utilize more information than just visual, such as the extension of the visual SLAM algorithms through the use of an IMU. However, in this thesis we confront some of these challenges by using just visual information and particularly by taking advantage of more geometric features than just points.

(a) Well-textured and well-illuminated narrow environment

(b) Low-textured and low-illuminated narrow environment

(c) Well-textured and well-illuminated large environment

(d) Low-textured and well-illuminated large environment

Figure 1.1: Environments with different texture and illumination. [4]

1.3 Our Contribution

In the present thesis we essentially embody the additional use of line segments into the stereo system of ORB-SLAM3[1] and present the ORB-LINE-SLAM system. The main goal of this work was to design an improved hybrid system in which the line segments would have an assistive role to the point features so as to elevate the robustness and efficacy of the original system in demanding conditions. The performance of point-based algorithms is inextricably connected with the capacity of the system to detect enough points in each frame, as well as determine adequate number of point matches so as to accurately estimate the pose of the camera. The main problem with this concept is that in cases that the environment is low-textured, or the frame is blurred (e.g., due to sharp moves of the camera), the point detection and matching algorithms fail to provide reliable results. A solution that we propose is to take advantage of more geometrical information of each frame. Particularly, by including line segment features, the performance of the system can be significantly enhanced in low-textured environments rich to line segments, as well as in difficult conditions, such as dark and blurred scenes, that even a small increase in the feature correspondences can play a vital role for the reliability of the estimation of the robot's pose. Another target of this thesis was to create the first open-source SLAM algorithm that depends exclusively on line segments and provide the community with a state-of-the-art solution that could be easily expanded by other researchers in the future.

Our system emerges from the combination of all the above ideas. In brief, the main contributions of this work are:

- A novel scheme of weighted fusion of point and line features through the use of an experimentally tuned adapting factor.
- The first open-source SLAM system that is able to function by using only line features.
- A detailed comparison of the accuracy of the most renowned line segment detectors and error functions regarding the SLAM problem.
- The incorporation of the line segments into the stereo algorithm of ORB-SLAM3.
- An open-source point-line system that outperforms the stereo version of ORB-SLAM3 which is currently the most accurate available stereo open-source implementation.

1.4 Organization

The present thesis is structured as follows:

- In Chapter 2, we analyze key terminology and mathematical background related to SLAM.
- In Chapter 3, we present some of the most relevant to our work and prominent systems in the field of SLAM in chronological order, and we briefly analyze some of their algorithmic characteristics.
- Chapter 4 relates to a detailed description of the structure and different components of ORB-LINE-SLAM, delving into the technical details of our system.
- In Chapter 5, we thoroughly examine the Bundle Adjustment problem for the feature-based SLAM and the different error functions used for the optimizations in our system.
- In Chapter 6, an in-depth evaluation of ORB-LINE-SLAM is presented, as well as a detailed comparison of the system with the most accurate methods available in the literature.
- Finally, in Chapter 7 we conclude with a discussion on the above and proposals for future extensions of this work.

Chapter 2

Key Terminology and Mathematical Background

In this chapter we explicate key terminology concerning SLAM and we develop the theoretical and mathematical background of concepts that will underpin our discussions in the next chapters. This chapter is dedicated to being a tutorial for the reader to understand the fundamental terms and ideas of SLAM so as to form a solid background for the rest of this thesis.

2.1 Definition of Fundamental Terms

Before we probe into the theoretical and mathematical background of core ideas of SLAM, let us first introduce some of the fundamental terms that will be constantly used in this work. The meaning of some terms is going to be adapted to reflect particular use cases of SLAM instead of their standard definitions.

A **Feature** is an individual characteristic about the content of an image; usually about if a specific area of the image has certain properties. Features may be particular structures in the image including points, lines, edges, or objects. In the case of SLAM, features on most occasions emerge as the result of algorithmic procedures in certain local regions in the image. There are many point and line detectors that has extensively been used during the last decades such as [83, 84, 85, 86, 87, 88, 89, 90, 91, 66, 67, 92], to name but a few. In the present thesis we will mostly examine point and line features and how they can be efficiently fused for the SLAM problem.

A **Feature Descriptor** is an algorithm which takes as input an image and outputs feature descriptors which are actually vectors. Feature descriptors encode interesting information, typically about the local neighborhood of a feature or the whole image, into a series of numbers and functions as a sort of numerical identity that can be used to match or distinguish one feature from another. Ideally, this information would be invariant under image transformation including changes in illumination, translation, scale, and in-plane rotation, so that the same feature can be detected again even if the image is transformed in some way. In general, descriptors can be categorized into two classes:

- **Local Descriptor:** It is a compact representation of a feature's local neighborhood. Local descriptors try to imprint the content of the pixels in a local neighborhood around a feature and hence are very appropriate for its representation regarding the procedure of matching. For this reason, they are widely used for the SLAM process. Examples of local descriptors have already been given as most of the previously mentioned works regarding feature detectors also present their corresponding descriptors.
- **Global Descriptor:** A global descriptor describes the whole image. They are generally more sensitive and not very robust compared to local descriptors as a

change in a region of the image affects all the image and thus the resulting descriptor. Global descriptors can be used in terms of SLAM for the loop detection process, where information from the whole image is necessary. Some examples of global descriptors are Shape Matrices, Invariant Moments (Hu, Zerinke), Histogram Oriented Gradients (HOG) and Co-HOG.

A **Stereo Camera** is a type of camera with two or more lenses with a separate image sensor or film frame for each lens [93]. The cameras are attached to a rigid body and the distance between them is stable. In this way, stereo cameras emulate human binocular vision, an attribute that gives them the ability of 3D perception. In essence, stereo cameras calculate the depth of a pixel usually in the left camera, by matching it with a corresponding pixel in the right camera and triangulating it in a 3D map using the baseline (distance between the cameras). Two key advantages of stereo cameras are the cost of components, and how it performs outdoors in bright areas in contrast to RGB-D cameras which get overpowered by the ambient lighting. On the other hand, due to the demanding process of matching, the computational cost of depth estimation is high, and additionally low light performance can be poor. However, most of the new stereo cameras carry an individual processor that enables onboard depth estimation and hence directly provides the user with a 3D point cloud. The accuracy of depth depends on three factors: resolution, lens focal length, and baseline. The wider the baseline and focal length are, the more accurate is the depth estimation at range, but concurrently the minimum distance in which depth can be determined elevates. Rectification is also another particularly useful procedure for stereo cameras, as rectified stereo images accelerate matching process and thus significantly lessen the computational cost of depth estimation.

A **Keyframe** is a frame in the video sequence that represents a visual summary and meaningful information about the scene. In visual SLAM, most of the times a frame is defined as keyframe if it comprises similar visual information (e.g., same point and line features) with other frames. By containing the most salient information about the scene, keyframes are mostly used for local and global optimizations which are extremely useful for SLAM systems as they give them feedback from previous frames of the video sequence in which the accumulated drift was less.

Bundle Adjustment is the task of simultaneous refining of the pose of the camera, the 3D position of the features that compose the map, and sometimes the optical parameters of the camera employed to acquire the images, given a set of images depicting a number of 3D points from different viewpoints. Bundle Adjustment constitutes a core component in most state-of-the-art SLAM systems and is typically invoked as a final refinement stage to rectify the position of the features in the reconstructed scene as well as a means for removing the cumulative drift. Specifically, it boils down to minimizing the

re-projection error between the image locations of observed and predicted image points, which is expressed as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms. Of these, Levenberg–Marquardt and Gauss–Newton have proven to be the most successful with the former to be the most preferable option in the field. The technical theory and mathematical background of both algorithms will be explained later in this chapter as well as bundle adjustment is going to be thoroughly examined in chapter 5.

Loop Closing is the ability of a robot to recognize whether, after an excursion of arbitrary length, it has returned to a previously visited area. Loop closure detection is of vital importance in the process of SLAM, as it helps to reduce the cumulative error of the robot's estimated pose and generate a consistent global map. Reliable loop closing is both essential and hard, and it is indisputably one of the greatest impediments to long term, robust SLAM as without it the accumulated drift would never be diminished. The optimization process related to loop closing is connected again to a nonlinear least-squares problem which is specifically solved through global bundle adjustment.

An **Inertial Measurement Unit (IMU)** is a sensor that typically consists of a combination of accelerometers, gyroscopes, and sometimes magnetometers so as to measure and report the velocity, acceleration, rotation, rotational rate and cardinal direction of a body. The use of an IMU concerning SLAM systems has been rapidly increasing the last years, as the self-motion information that it provides can be critical in many difficult occasions where the visual information is not adequate.

2.2 Mathematical Background of Core Concepts

2.2.1 Camera Calibration

Concerning SLAM, **Camera Calibration** essentially refers to the estimation of parameters regarding the lens and image sensor of a camera in order to complete operations with specified performance measurements. These parameters are called **Intrinsic or Internal Parameters** and are responsible for mapping between pixel and camera coordinates in the image frame. Such parameters typically are optical center, focal length, and radial distortion coefficients of the lens [94].

In general, the problem of camera calibration aims to the computation of the camera matrix which maps the 3D world scene into the image plane. Except for the intrinsic parameters, the determination of the camera matrix requires the estimation of the so-called **Extrinsic or External Parameters**. These parameters describe the translation and rotation of the frame of the camera with respect to some world coordinate system; i.e., they are constantly estimated through the localization process of the SLAM system.

There are two different types of cameras that are typically used for SLAM; the **pinhole** camera and the **fisheye** camera. Their main distinction is that the fisheye lens attached to the fisheye camera accounts for high distortion in contrast to the pinhole camera which ideally does not even have a lens. In this section, we will focus on the pinhole camera model as all the experiments in this thesis have been committed with this type of camera, not mentioning the fact that it constitutes the most popular camera model for SLAM.

A pinhole camera is a basic camera model that does not include a lens. However, to accurately simulate a genuine pinhole camera, in the camera model it is also incorporated radial and tangential lens distortion. The operation of a pinhole camera is quite simple and it mimics human vision. Particularly, light rays that reflect from the 3D scene pass through the aperture and project an inverted image on the opposite side of a light-proof box inside the camera which has small hole in one side. This procedure is known as the camera obscura effect. Then, in order to visualize the image correctly the camera just places upright the image of the scene in the virtual image plane [95]. The operation of the camera model is briefly depicted in Figure 2.1.

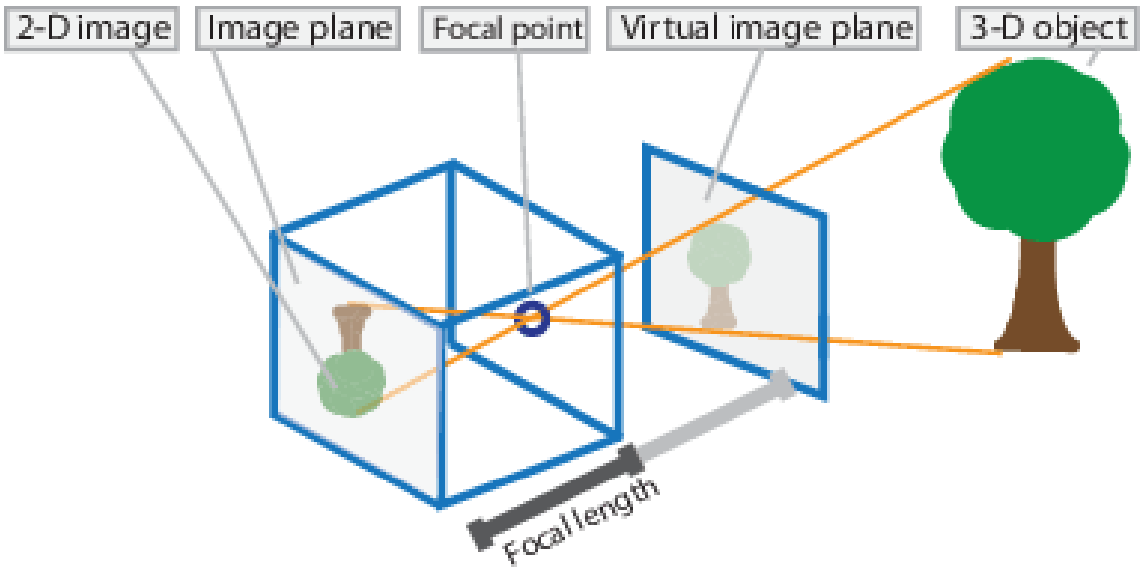


Figure 2.1: Operation of pinhole camera model. [5]

If a 2D point in the image is denoted by $m = [x, y]^T$ and a 3D point in the world by $M = [X, Y, Z]^T$, then their corresponding homogeneous vectors can be defined as: $\widetilde{m} = [x, y, 1]^T$ and $\widetilde{M} = [X, Y, Z, 1]^T$. The usual pinhole model that gives the relationship between a 3D point M and its image projection m is

$$s\widetilde{m} = AP\widetilde{M}, \tag{2.1}$$

where s is an arbitrary scale factor, P the pose of the camera in respect of the world coordinate system, i.e., the extrinsics, and A the intrinsics of the camera. The intrinsics

and extrinsics parameters can be explicitly defined as:

$$A = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \; and \; P = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

with (f_x, f_y) the focal length in pixels multiplied by corresponding scaling factors in x and y axes which are usually 1, (c_x, c_y) the coordinates of the principal point, and γ the parameter describing the skewness of the two image axes which is typically considered to be zero in the case of pinhole cameras. So, the camera calibration problem can be solved by determining the camera matrix C which is denoted by $C = AP$.

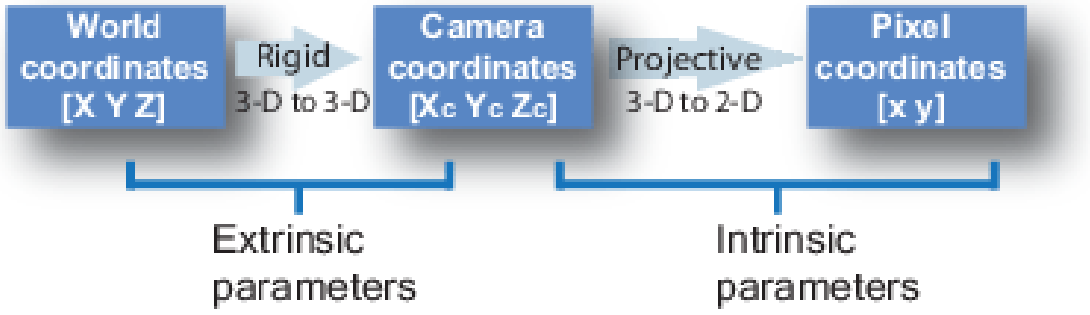


Figure 2.2: Camera calibration coordinates blocks. [5]

Considering that matrix P is known from the localization process we only need to estimate the intrinsics of the camera. The focal length can be defined as the distance between the center of the lens and the sensor, the so-called focal point, while the principal point is the point on the image plane onto which the perspective center is projected (Figure 2.3).

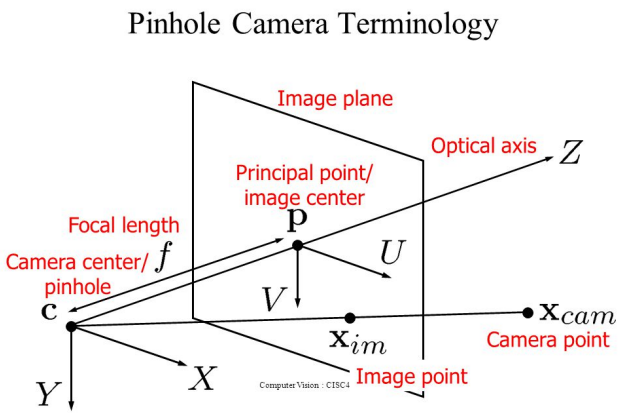


Figure 2.3: Pinhole camera terminology. [6]

Up to now, we have not taken into consideration that although the pinhole camera model ideally does not include distortion coefficients, some pinhole cameras introduce distortion to images. Generally, distortion can be divided into radial and tangential distortion. Radial distortion is the inability of the lens to be rectilinear, i.e., it images straight lines into curved, while tangential distortion occurs when the lens and the image plane are not parallel.

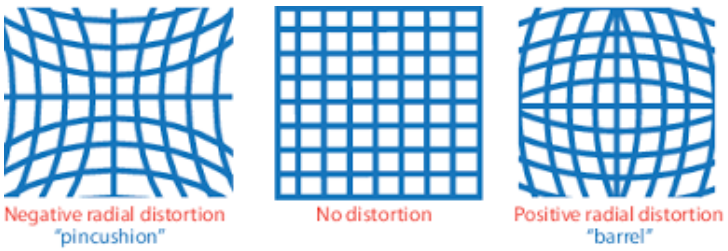


Figure 2.4: Radial distortion. [5]

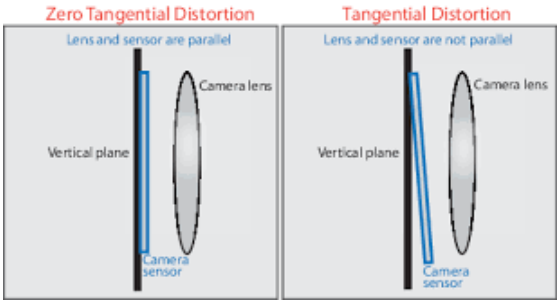


Figure 2.5: Tangential distortion. [5]

The mathematical modeling of the equations regarding the radial coefficients was first presented in [96, 94]. If $(x_{distorted}, y_{distorted})$ are the the real (distorted) and (x, y) the ideal (undistorted) normalized image coordinates, then the radial distortion can be expressed as follows:

$$x_{distorted} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6), \tag{2.3}$$
$$y_{distorted} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6), \tag{2.4}$$

and as for the tangential distortion:

$$x_{distorted} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)], \tag{2.5}$$
$$y_{distorted} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y], \tag{2.6}$$

where k_1, k_2 and k_3 stand for the radial distortion coefficients of the lens, p_1 and p_2 for the tangential distortion coefficients of the lens, and r^2 for the sum of the squares of x and y, i.e., $r^2 = x^2 + y^2$.

We will not proceed to the explicit solution regarding the camera calibration parameters as the purpose of this thesis is not to be a tutorial for camera calibration. There have been developed many algorithms that use different techniques to solve this problem such as [96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108], to cite a few. Nevertheless, today the most dominant method seems to be the one illustrated in [109], not mentioning that also many automated software solutions have been advanced for fast camera calibration, like by using Matlab [110], or OpenCV [68].

2.2.2 Image Rectification

Image Rectification is a transformation process used to re-project image planes onto a common plane parallel to the line that intersects optical centers. Specifically, image rectification is the process of forcing the epipolar lines to be parallel to the horizontal axis just as depicted in Figure 2.6.

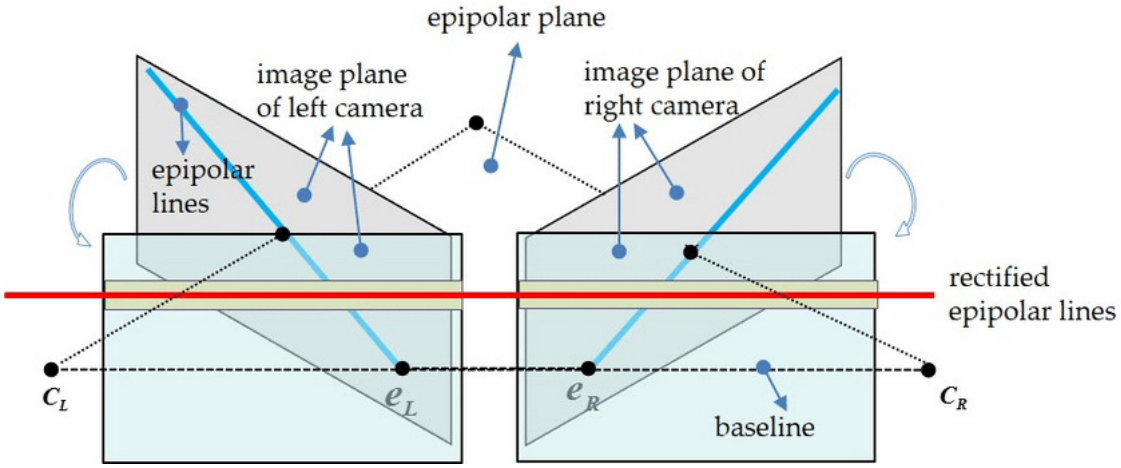


Figure 2.6: Schematic diagram of stereo rectification. [7]

In terms of SLAM, the utility of image rectification in stereo vision is to simplify the stereo matching problem (i.e., the matching of points between the two stereo images) and as a result decrease the computational cost for the estimation of depth. Particularly, by using rectified images, in order to precisely match a pixel in the left image, someone needs to search only in a local neighborhood in the right image according to the position of the pixel that we are seeking to match and the baseline of the camera, and not in the whole image. So, by assuming that each pixel is described by a local descriptor, we just need to compare the descriptor of the pixel that we are seeking to match with the descriptors of the pixels in the corresponding neighborhood (ideally in the common epipolar line). In this way, not only the computations are significantly reduced as the number of comparisons abates, but also the efficacy of the algorithm is enhanced as many outliers that would incorrectly constitute correspondences are now discarded because they do not belong in the region of interest.

We will now formally formulate the image rectification problem according to [8]. If the frames of a pair of pinhole cameras in 3D space are denoted by C and C', and m and m' are the projections through these cameras of a 3D point M in the images I and I' respectively, then the epipolar constraint can be defined as:

$$m'^T F m = 0, \tag{2.7}$$

where F is the so-called fundamental matrix. The above are clearly depicted in Figure 2.7.

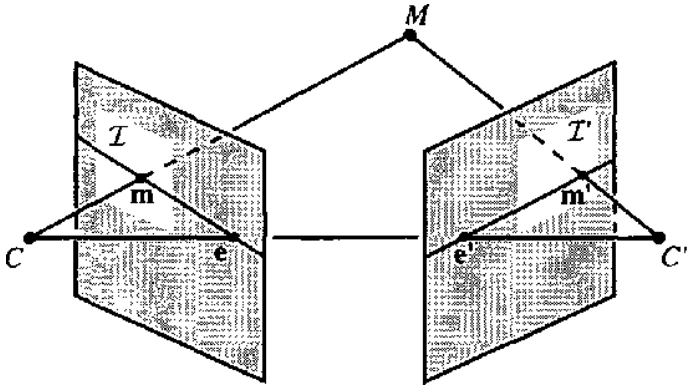


Figure 2.7: Epipolar geometry between a pair of images. [8]

To rectify the images, we need to apply a homography to each one of them that maps the epipole with a predetermined point. We assume that this point is $i = [1\ 0\ 0]^T$ (a point at ∞) and the corresponding fundamental matrix is denoted by:

$$\widetilde{F} = [\ i\ ]_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{2.8}$$

where the notation $[\ i\ ]_x$ denotes the antisymetric matrix representing the cross product with i. Now if H and H' are the homography matrices to be applied to images I and I' respectively, then the corresponding rectified image points $\widetilde{m}$ and $\widetilde{m'}$ defined

$$\widetilde{m} = Hm \quad \text{and} \quad \widetilde{m'} = H'm'. \tag{2.9}$$

So, from the equations (2.7) and (2.9) it follows that:

$$\widetilde{m'^T}\widetilde{F}\widetilde{m} = 0, \tag{2.10}$$

$$m'^T H'^T \widetilde{F} H m = 0. \tag{2.11}$$

And now from the equations (2.7), (2.8), and (2.11) it can be concluded that:

$$F = H'^T \widetilde{F} H = H'^T [\ i\ ]_x H. \tag{2.12}$$

The image rectification problem refers to the computation of a pair of homographies H and H' that minimize image distortion as individually they are not unique. Homography matrices H and H' can be denoted by:

$$H = \begin{bmatrix} \alpha^T \\ \beta^T \\ \gamma^T \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix}, \tag{2.13}$$

$$H' = \begin{bmatrix} \alpha'^T \\ \beta'^T \\ \gamma'^T \end{bmatrix} = \begin{bmatrix} \alpha'_1 & \alpha'_2 & \alpha'_3 \\ \beta'_1 & \beta'_2 & \beta'_3 \\ \gamma'_1 & \gamma'_2 & \gamma'_3 \end{bmatrix}. \tag{2.14}$$

Similarly, to the previous subsection we will not proceed to the explicit solution. There have been developed many techniques for the estimation of the rectification homographies including [8, 111, 112], to name a few. Today, such algorithms are mostly implemented in programming platforms and used in an automated way so as to rectify stereo images.

Figure 2.8: Example of stereo rectified images. [9]

2.2.3 Triangulation

Triangulation is the process which aims to the determination of the coordinates of a point in 3D space given its position onto two, or more, image planes. Essentially,

triangulation refers to the estimation of depth of a common point in two stereo images. Supposing that p_1 denotes a point in the left stereo image that we want to project onto 3D space, p_2 is its corresponding point in the right image, b the baseline of the camera, f the focal length, and that the position of the left camera is known from the extrinsics (through the localization process), then the triangulation problem can be solved quite easily by using simple geometry.

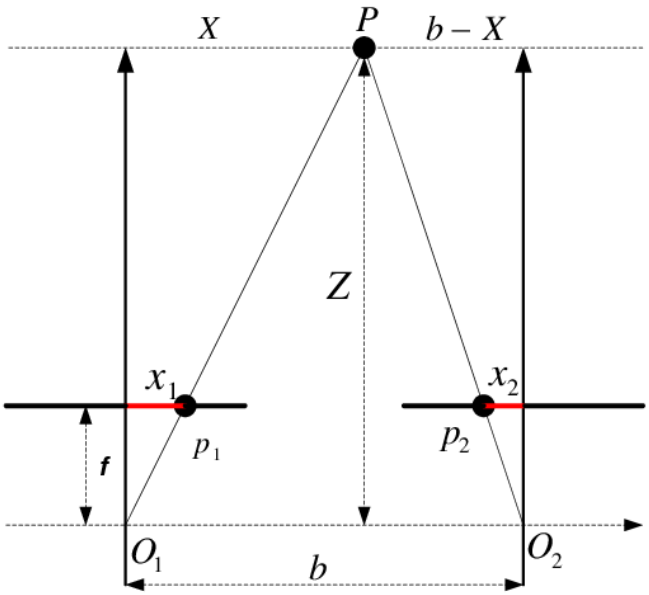


Figure 2.9: Triangulation scheme of stereo vision. [10]

From the scheme depicted in Figure 2.9 by using similar triangles we can effortlessly formulate the following mathematical relationships:

$$\frac{X}{Z} = \frac{x_1}{f}, \tag{2.15}$$

$$\frac{b - X}{Z} = \frac{x_2}{f}. \tag{2.16}$$

And now from equations 2.15 and 2.16 disparity can be denoted by

$$d = x_1 - x_2 = \frac{bf}{Z}. \tag{2.17}$$

Hence depth Z is inversely proportional to disparity and can be easily computed through the equation $Z = \frac{bf}{x_1 - x_2}$. As for the coordinates of the 3D point on x and y axes, they considered known from the position of the camera, the position of the point in the image plane and the scaling factor.

2.2.4 Gauss-Newton Algorithm

Gauss-Newton method is used in SLAM for solving the nonlinear least-squares problem connected to bundle adjustment. Its purpose is to determine a local minimum of a nonlinear function. This algorithm was first presented in 1809 in the work "Theoria motus corporum coelestium in sectionibus conicis solem ambientum", and it is named after the mathematicians Carl Friedrich Gauss and Isaac Newton. We will explicitly analyze the mathematics of this method as it is used in the most crucial part of SLAM, the estimation of the pose of the camera, and so the user needs to know its technical details. The reader can find further information in [113].

Assuming that there are m error functions $r = (r_1, ..., r_m)$ that refer to the re-projection errors, and n variables $\beta = (\beta_1, ..., \beta_n)$ with $m \geq n$ that refer to the pose of the camera and sometimes to the position of the points used for the optimization, the Gauss-Newton method iteratively estimates the value of the variables $\beta_1, ..., \beta_n$ that minimize

$$\boldsymbol{S}(\beta) = \sum_{i=1}^{m} r_i(\beta)^2. \tag{2.18}$$

To solve the problem, firstly we need to do an initial guess $\beta^{(0)}$, usually in SLAM the previous pose of the camera and position of the points, that is of pivotal importance regarding the convergence of the method to a local minimum. Next, the algorithm proceeds by the iterations

$$\beta^{(s+1)} = \beta^{(s)} - (\boldsymbol{J}_r^T \boldsymbol{J}_r)^{-1} \boldsymbol{J}_r^T r(\beta^{(s)}), \tag{2.19}$$

where, if r and β are column vectors, the Jacobian matrix is denoted by

$$(\boldsymbol{J}_r)_{ij} = \frac{\partial r_i(\beta^{(s)})}{\partial \beta_j}. \tag{2.20}$$

Now, if $\Delta = \beta^{(s+1)} - \beta^{(s)}$, then from (2.19) it can be concluded that

$$\Delta = -(\boldsymbol{J}_r^T \boldsymbol{J}_r)^{-1} \boldsymbol{J}_r^T r(\beta^{(s)}), \tag{2.21}$$

$$\boldsymbol{J}_r^T \boldsymbol{J}_r \Delta = -\boldsymbol{J}_r^T r(\beta^{(s)}). \tag{2.22}$$

By supposing that $A = \boldsymbol{J}_r^T \boldsymbol{J}_r$, $x = \Delta$, and $b = -\boldsymbol{J}_r^T r(\beta^{(s)})$, equation (2.22) turns into the conventional matrix equation of form $Ax = b$, which can be easily solved in many ways.

2.2.5 Levenberg–Marquardt Algorithm

Similarly to Gauss-Newton, **Levenberg–Marquardt** algorithm (LMA or just LM) is concerned with solving nonlinear least-squares problems. This method took its name from Kenneth Levenberg and Donald Marquardt that published the algorithm in 1944 and 1963, respectively. Levenberg-Marquardt method is a hybrid version of Gauss-Newton

and gradient descent as it takes advantage of the benefits of both. A detailed description can be found in [114]. By having already formulated the nonlinear least-squares problem, the only contribution of Levenberg's to Gauss-Newton method is the addition of a damping factor in the equation (2.22) which is now expressed as follows:

$$(\boldsymbol{J}_r^T \boldsymbol{J}_r + \boldsymbol{\lambda} \boldsymbol{I})\Delta = -\boldsymbol{J}_r^T r(\beta^{(s)}), \tag{2.23}$$

where $\boldsymbol{\lambda}$ is the non-negative damping factor and $\boldsymbol{I}$ the identity matrix.

Parameter $\boldsymbol{\lambda}$ is modified at each iteration according to the change in the cost function. Specifically, if $\boldsymbol{S}$ swiftly reduces, then $\boldsymbol{\lambda}$ takes smaller values so that the method approaches Gauss-Newton, while in the case that the value of $\boldsymbol{S}$ is slightly altered, $\boldsymbol{\lambda}$ sharply increases and the algorithm gets closer to gradient-descent direction. Geometrically, the main difference between gradient descent and Gauss-Newton is that the first calculates only the first derivative and accordingly takes a step to its direction, while the second, except for the slope, is also concerned with the curvature which represents the second derivative. Specifically, Gauss-Newton takes small steps if the curvature is high and big ones if the curvature is low. Although this method usually converges faster compared to gradient descent, it can be very unstable in cases where the initial guess is far from a local minimum. In such circumstances, Gauss-Newton algorithm converges particularly slowly or even not at all. Consequently, Levenberg–Marquardt algorithm by imitating gradient descent when it is far from a minimum, and Gauss Newton when it is closer to a minimum, constitutes simultaneously a robust and efficient solution as it can converge to a minimum in a reasonable number of iterations even if the initial guess is far from it. For this reason, Levenberg–Marquardt method is regularly the preferable choice concerning the field of SLAM. On the other hand, Gauss-Newton is mostly preferred for problems associated with well-behaved functions and reasonable starting parameters since in these cases it converges faster than Levenberg–Marquardt.

Chapter 3

Related Work

This chapter is dedicated to presenting some of the most notable works in the field of SLAM. Beginning from the original approaches which caught the eye of the scientific community in the past, and progressing to the recent state-of-the-art algorithms, we analyze their technical details by giving emphasis to common characteristics with our work.

3.1 Probabilistic SLAM

3.1.1 Bayes Filter

The *Bayesian filter* is a recursive algorithm for estimating the belief distribution *bel* which is associated with the pose of the robot, given a sequence of measurements and control signals. Hence, the *Bayesian filter* is only used for localization. Assuming that at time t the measurements data are denoted by $\mathbf{z}_t$, the control signals by $\mathbf{u}_t$, and the pose of the robot by $\mathbf{x}_t$, then to estimate the belief $bel(\boldsymbol{x}_t)$ two steps are performed: the prediction step, and the measurement update step. In the prediction step, the algorithm applies the control $\mathbf{u}_t$ to get the transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$, while in the measurement update step, the algorithm multiplies the belief $\overline{bel}(\mathbf{x}_t)$ with the probability associated to the observance of the measurement $\mathbf{z}_t$, and then normalizes this product with a constant η as in general this product is not a probability. According to [11] the *Bayesian filter* is stated as follows:

Algorithm 3 Bayesian Filter Algorithm [11]

Input $bel(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$
for all $\mathbf{x}_t$ **do**
  $\overline{bel}(\mathbf{x}_t) = \int \mathbb{P} = [\mathbf{x}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t]\ bel(\mathbf{x}_{t-1})\ \partial \mathbf{x}_{t-1}$
  $bel(\mathbf{x}_t) = \eta \cdot \mathbb{P} = [\mathbf{z}_t \mid \mathbf{x}_t]\ \overline{bel}(\mathbf{x}_t)$
end for
return bel$(\mathbf{x}_t)$

3.1.2 Kalman Filter

The *Kalman Filter* is a modified version of the *Bayes Filter*, where it is hypothesized that the beliefs are represented by multivariate normal distributions. The multivariate normal distribution can be described by the following equation

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}}\ exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}. \tag{3.1}$$

For the state belief *bel(*$\boldsymbol{x}_t$*)* to be Gaussian distributed the *Kalman Filter* assumes that both system and observation model equations are linear. Particularly, the state of the robot and the measurements must be modeled as:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t, \tag{3.2}$$

$$z_t = C_t x_t + \delta_t, \tag{3.3}$$

where ε_t, δ_t represent Gaussian noise, and if we suppose that n, m, and k are the dimensions of the state vector, control signals, and measurements, respectively, then A_t, B_t, and C_t are their corresponding matrices with size $n \times n$, $n \times m$, $k \times n$, respectively. By multiplying the vectors concerning the state, control, and measurements with their corresponding matrices, it is apparent that equations 3.2 and 3.3 become linear. By respectively incorporating the equations 3.2 and 3.3, in 3.1, it can be concluded that the state transition probability $p(x_t \mid u_t, x_{t-1})$, and the measurement probability $p(z_t \mid x_t)$ are normally distributed. Finally, the *Kalman Filter* assumes that the initial belief *bel(*$\boldsymbol{x}_0$*)* is normally distributed and can be explicitly expressed as:

$$bel(\mathbf{x}_0) = p(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}}\ exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\}, \tag{3.4}$$

where Σ_0 is the covariance and μ_0 is the mean value of this belief. By using the assumptions that the state transition probability, the measurement probability, and the initial belief are normally distributed, it is proved in [11] that the posterior belief *bel(*$\boldsymbol{x}_t$*)* is always a Gaussian.

The *Kalman Filter* algorithm follows the same strategy as the *Bayes Filter*. If the posterior *bel(*$\boldsymbol{x}_t$*)* at time t is represented by the mean μ_t and the covariance Σ_t, and R_t, Q_t are the covariance matrices associated with ε_t and δ_t, respectively, then the *Kalman Filter* algorithm is described as follows:

Algorithm 4 Kalman Filter Algorithm [11]

Input $(\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$

1: $\overline{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
2: $\overline{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
3: $K_t = \overline{\Sigma}_t\ C_t^T\ (C_t\ \overline{\Sigma}_t\ C_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t\ (z_t - C_t\ \overline{\mu}_t)$
5: $\Sigma_t = (I - K_t\ C_t)\ \overline{\Sigma}_t$
6: **return** $\mu_t,\ \Sigma_t$

Accordingly to *Bayes Filter* the prediction step corresponds to the first two lines of the algorithm, while the measurement update step to lines 3, 4 and 5. The only contrast that is discrete between the two can be observed in line 4 where the kalman gain K_t is

calculated. Qualitatively, the kalman gain refers to the degree to which the measurement will affect the final posterior.

As it was previously mentioned, in order for the posteriors to be Gaussian distributed, the *Kalman Filter* assumes that the associated equations are linear. If nonlinear equations were applied, then the resulting distributions would not be Gaussian. Hence, *Kalman Filter* is only effective for linear systems. However, the process of localizing a robot is generally not a linear problem. A nonlinear version of the *Kalman Filter* is the *Extended Kalman Filter* which is going to be presented later in this chapter.

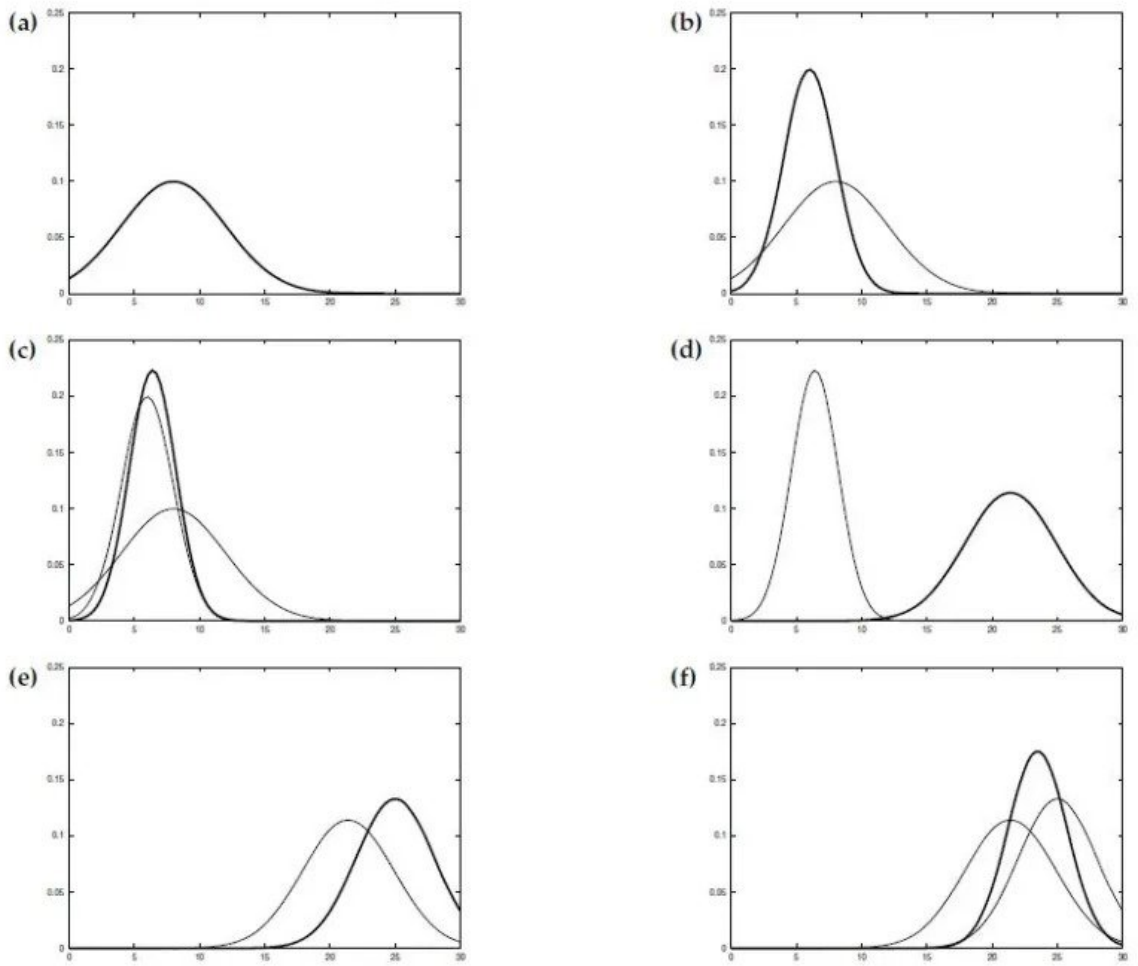


Figure 3.1: Illustration of the conversion of the initial belief after applying measurement and control data [11]. (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

3.1.3 Particle Filter

In essence, the *Particle Filter* is a nonparametric form of the *Bayes Filter*. In its case, the posterior *bel(*$\boldsymbol{x}_t$*)* is estimated by drawing samples from this posterior. Hence, contrary to the *Bayes* and *Kalman Filters* which in order to determine posteriors assume that the distributions are Gaussian, the *Particle Filter* can deal with a much broader space of distributions and thus with problems that are not linear.

The basic distinction of the *Particle Filter* is the resampling step. In this step, in order to approximate a density function f called the target distribution that is connected to the belief *bel(*$\boldsymbol{x}_t$*)*, the *Particle Filter* takes samples from a related density function g called the proposal distribution, which represents the belief $\overline{bel}(\mathbf{x}_t)$. The transition from the proposal to the target distribution is made by weighting the extracted samples by the quotient

$$w^{[m]} = \frac{f(x^{[m]})}{g(x^{[m]})}, \tag{3.5}$$

where the samples or the so-called particles are denoted

$$X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}, \ with\ 1 \leqslant m \leqslant M. \tag{3.6}$$

The resampling step is clearly illustrated in Figure 3.2.

The algorithm of the *Particle Filter* follows the same logic as the *Bayes Filter* with the difference that the belief *bel(*$\boldsymbol{x}_t$*)* is now replaced by the particles X_t, and it can be described as follows:

Algorithm 5 Particle Filter Algorithm [11]

Input $(\mathrm{X}_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$
$\overline{X_t} = X_t = \emptyset$
for m = 1 to M **do**
  sample $x_t^{[m]} \sim p(\mathbf{x}_t \mid \mathbf{u}_t, \mathrm{x}_{t-1}^{[m]})$
  $w_t^{[m]} = p(\mathbf{z}_t \mid x_t^{[m]})$
  $\overline{X_t} = \overline{X_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
end for
for m = 1 to M **do**
  draw i with probability $\propto w_t^{[i]}$
  add $x_t^{[i]}$ *to* X_t
end for
return X_t

In the field of SLAM, particle filters are mostly used for localization through the use of the *Monte Carlo Localization Algorithm* [115, 116].

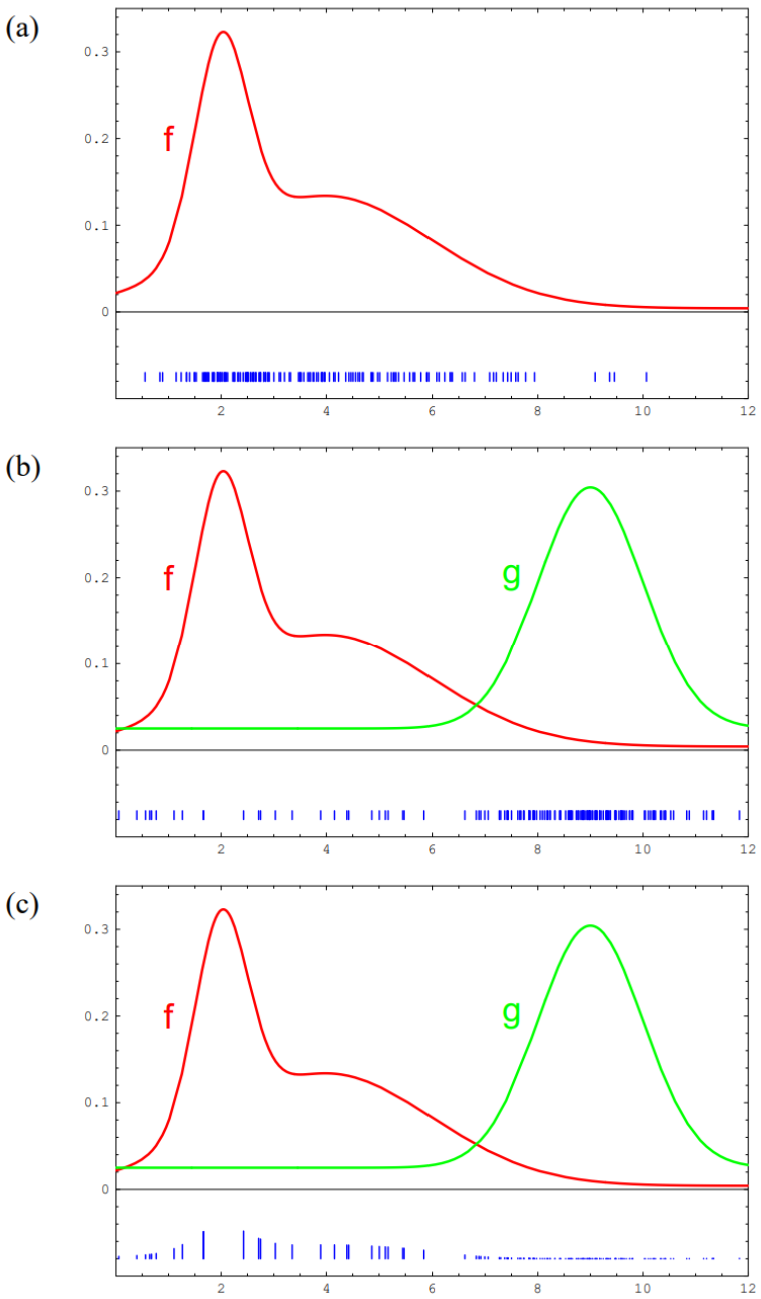


Figure 3.2: Illustration of resampling step [11]. (a) target distribution f and how it would be to generate samples from this function (blue lines at the bottom of the diagram), (b) target and proposal distributions f and g, respectively, and samples generated from g, (c) weighted samples according to the factor $w^{[m]}$.

3.1.4 SLAM with the Extended Kalman Filter

The *Extended Kalman Filter* is an extended version of the previously described *Kalman Filter* which focuses on nonlinear systems where the states of the system are described by multimodal distributions. Following the same strategy as the *Kalman Filter*, and by assuming that N_{t-1} is the momentary size of the map, the algorithm of the *Extended Kalman Filter SLAM* is stated in the next page as follows:

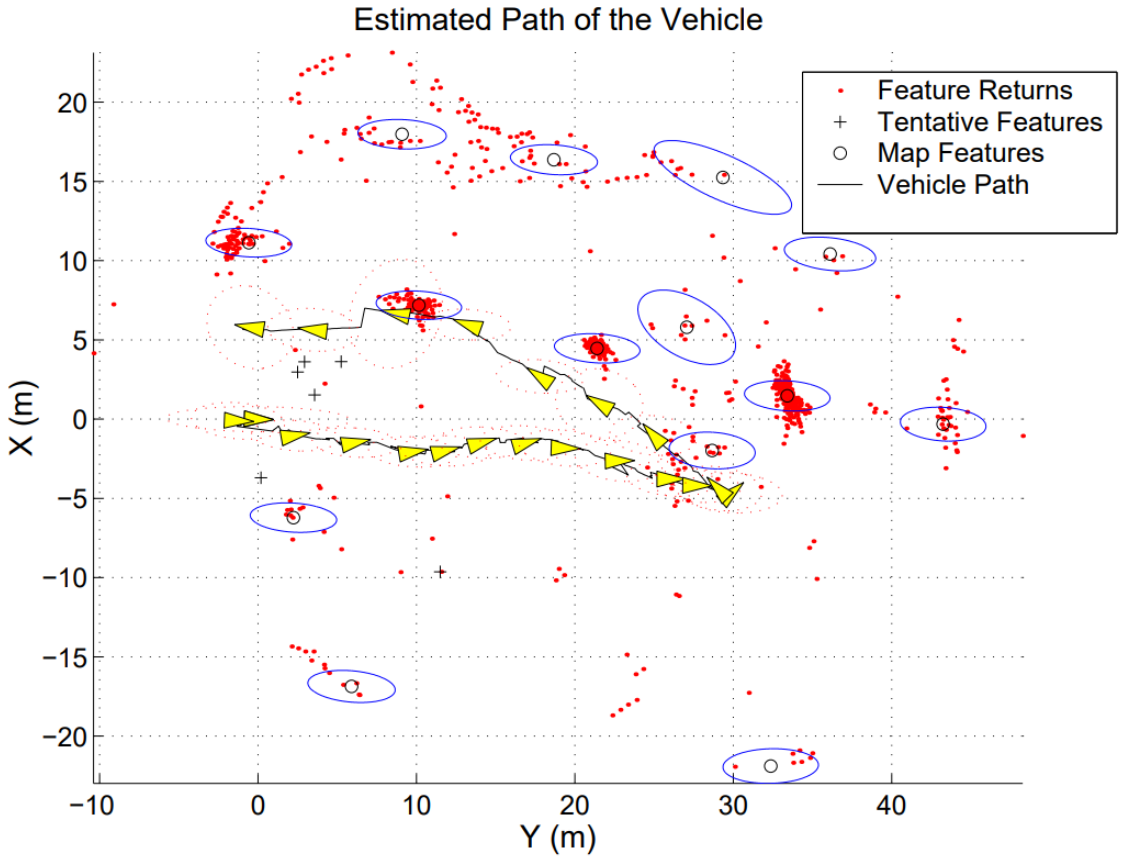


Figure 3.3: Example of Extended Kalman Filter estimation of the map and the vehicle pose [11].

Similar to the *Kalman Filter*, the *Extended Kalman Filter* for SLAM includes the prediction and update steps. The prediction step is clearly visible in lines 2 through 5 where the initial posterior is updated through the motion model, while the update step takes place in lines 23 and 24. The main distinction of this algorithm lies in measurement update loop. Starting in line 8, a new landmark's location is initialized according to the pose of the robot and the measurement of the sensor. Then, from line 9 through 17

Algorithm 6 Extended Kalman Filter SLAM Algorithm [11]

Input $(\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t,\ \mathbf{z}_t,\ N_{t-1})$

$N_t = N_{t-1}$

$F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 \end{pmatrix}$

$\overline{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{u_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{u_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ \frac{u_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ \omega_t\Delta_t \end{pmatrix}$

$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & \frac{u_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ 0 & 0 & \frac{u_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{u_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta_t) \\ 0 & 0 & 0 \end{pmatrix} F_x$

$\overline{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + F_x^T\ R_t\ F_x$

$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ **do**

$\begin{pmatrix} \overline{\mu}_{N_t+1,x} \\ \overline{\mu}_{N_t+1,y} \\ \overline{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ sin(\phi_t^i + \overline{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$

for $k = 1$ to $N_t + 1$ **do**

$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{k,x} - \overline{\mu}_{t,x} \\ \overline{\mu}_{k,y} - \overline{\mu}_{t,y} \end{pmatrix}$

$q_k = \delta_k^T \delta_k$

$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \text{atan2}(\delta_{k,y}, \delta_{k,x}) - \overline{\mu}_{t,\theta} \\ \overline{\mu}_{k,s} \end{pmatrix}$

$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 & 0 & 0 & 0 & 0...0 \\ & & & & & & & \\ 0 & 0 & 0 & 0...0 & 1 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 1 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 0 & 1 & 0...0 \end{pmatrix}$

$H_t^k = \frac{1}{q_k} \begin{pmatrix} \sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & -\sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} & 0 \\ \delta_{k,y} & \delta_{k,x} & -1 & -\delta_{k,y} & -\delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$

$\Psi_k = H_t^k\ \overline{\Sigma}_t\ [H_t^k]^T + Q_t$

$\pi_k = (z_t^i - \hat{z}_t^k)\ \Psi_k^{-1}\ (z_t^i - \hat{z}_t^k)$

end for

$\pi_{N_t+1} = \alpha$

$j(i) = \arg\min_k \pi_k$

$N_t = \max(N_t, j(i))$

$K_t^i = \overline{\Sigma}_t\ [H_t^{j(i)}]^T\ \Psi_{j(i)}^{-1}$

end for

$\mu_t = \overline{\mu}_t + \sum_i K_t^i\ (z_t^i - \hat{z}_t^{j(i)})$

$\Sigma_t = (I - \sum_i K_t^i\ H_t^{j(i)})\ \overline{\Sigma}_t$

return $\mu_t,\ \Sigma_t$

various quantities regarding all the landmarks (including the new one) are updated, while in line 18 the threshold for the creation of a new landmark is established. If the Mahalanobis distance to all the landmarks of the map exceeds this threshold, the new landmark is accepted. In lines 19 and 20 the algorithm determines the correspondence through maximum likelihood estimation and afterwards increments the landmark counter if the correspondence is associated with a previously unseen landmark. Finally, the Kalman gain is computed in line 21.

Historically, the *Extended Kalman Filter* SLAM algorithm constitutes the earliest, and probably the most influential SLAM algorithm ever developed. However, subsequently more probabilistic formulations for SLAM have been established that caught the eye of the community. The most celebrated one is the *Rao-Blackwellised Particle Filter*, which incorporates the Rao–Blackwell theorem to enhance the sampling process of the classical *Particle Filter*. The first works that implemented the *Rao-Blackwellised Particle Filter* in the process of SLAM are presented in [16] and [17].

3.2 LIDAR SLAM

3.2.1 LOAM: Lidar Odometry and Mapping in Real-time

The *LOAM* [2] algorithm has been indisputably one of the hottest topics in the field of 3D LiDAR SLAM the last decade. The core of the algorithm is quite similar to the feature-based methods of visual SLAM as it relies on estimating the pose of the LiDAR by minimizing the overall distance of feature correspondences. The odometry part of *LOAM* runs at the high frequency of 10 times per sweep (10Hz). Specifically, every time the LiDAR generates a point cloud, the feature points go through a 3-level filtering so as to be selected for the matching process. In the first level, in order to separate evenly the environment, the scan is divided into 4 identical subregions in each of which at most 2 edge and 4 planar points can be extracted. Edge and planar points are picked out according to whether their c value, which represents the smoothness of the local surface, exceeds or is smaller than an established threshold. After a point is selected, the algorithm checks that none of its surrounding points is already chosen, and that the angle between the surface patch that it belongs and the laser beam exceeds an established threshold, or that it is on boundary of an occluded region (Figure 3.4).

The pose of the LiDAR is estimated within every sweep and so are the feature correspondences. Assuming that $\overline{P}_k$ is the reprojected point cloud of the k sweep to the time stamp of the k+1 sweep, and that P_{k+1} is the point cloud of the k+1 sweep, then the algorithm tries to match edge and planar points between $\overline{P}_k$ and P_{k+1}. Specifically, if $\overline{E}_{k+1}$ and $\overline{H}_{k+1}$ are the reporjected edge and planar points to the beginning of the sweep according to the recursively estimated 6-DOF motion during this sweep, the algorithm determines the closest neighbors of $\overline{E}_{k+1}$ and $\overline{H}_{k+1}$ in $\overline{P}_k$.

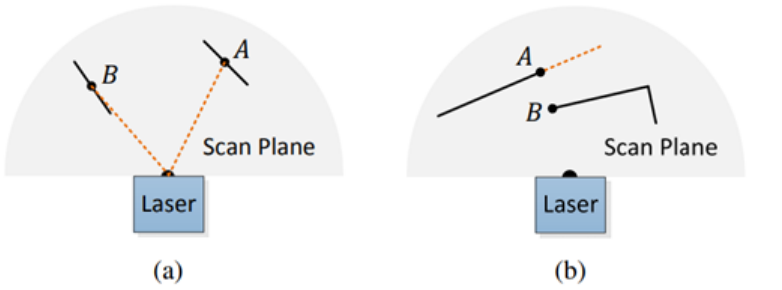


Figure 3.4: (a) The solid line segments represent local surface patches. Point A is on a surface patch that has an angle to the laser beam (the dotted orange line segments). Point B is on a surface patch that is roughly parallel to the laser beam. B is considered an unreliable laser return and it is not selected as a feature point. (b) The solid line segments are observable objects to the laser. Point A is on the boundary of an occluded region (the dotted orange line segment), and can be detected as an edge point. However, if viewed from a different angle, the occluded region can change and become observable. Point A is not treated as a salient edge point or selected as a feature point [2].

Regarding the edge points, if i is an edge point in $\overline{E}_{k+1}$ and j is its closest neighbor in $\overline{P}_k$, then the edge line emerges from the edge points j, and l, which is the closest neighbor of i in the two consecutive scans to the scan of j. To estimate the pose of the LiDAR, the algorithm seeks to minimize the point to line distance between i and the aforementioned edge line, which particularly can be described as:

$$d_\varepsilon = \frac{|(\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k,j)}) \times (\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k,l)})|}{|\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}|}, \tag{3.7}$$

where $\overline{X}^L_{(k+1,i)}, \overline{X}^L_{(k,j)}$, and $\overline{X}^L_{(k,l)}$ are the coordinates of i, j, and l in the LiDAR coordinate system $\{L\}$.

As for the planar points, if i is a planar point in $\overline{H}_{k+1}$ and j is its closest neighbor in $\overline{P}_k$, then the planar patch emerges from the points j, l, and m, where l and m are the closest neighbors of i, one in the same scan of j, and the other in two consecutive scans to the scan of j. Similarly to the edge points, now the point to plane distance can be denoted by:

$$d_H = \frac{\left| \begin{array}{c} (\overline{X}^L_{(k+1,i)} - \overline{X}^L_{(k,j)}) \\ ((\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}) \times (\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,m)})) \end{array} \right|}{|(\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,l)}) \times (\overline{X}^L_{(k,j)} - \overline{X}^L_{(k,m)})|}, \tag{3.8}$$

where $\overline{X}^L_{(k,m)}$ is the coordinates of m in $\{L\}$.

The motion estimation problem yields to determining the T_{k+1}^{L} LiDAR pose transform between $[t_{k+1}, t]$ by minimizing the distances 3.7 and 3.8. $LOAM$ approaches the problem through a robust fitting method based on [117], and particularly by assigning a bisquare weight to each correspondence. The larger the distance is of the feature point from its correspondence, the smaller is the weight assigned to this correspondence, and if this distance exceeds an established threshold, then the correspondence is considered as an outlier and it is assigned with zero weight. The problem is solved through nonlinear iterations by using the Levenberg–Marquardt Algorithm.

The LiDAR mapping algorithm runs at a lower frequency and specifically only once per sweep (1Hz), and it essentially optimizes the estimation of the odometry algorithm regarding the motion of the LiDAR. Having already calculated the motion of the LiDAR during the sweep through the odometry algorithm, the mapping algorithm projects the point cloud of this sweep to the map, and searches for correspondences between this cloud and the cloud of the map. By using the same strategy as in the odometry algorithm, the mapping part minimizes the distance between edge points and lines and planar points and patches through the Levenberg–Marquardt Algorithm so as to optimize again the motion of the LiDAR. The map clouds produced by the system, and in general by the LiDAR SLAM systems, seem remarkably precise and accurate which is a salient advantage compared to the visual SLAM systems which may include undesirable outliers that adulterate the map.

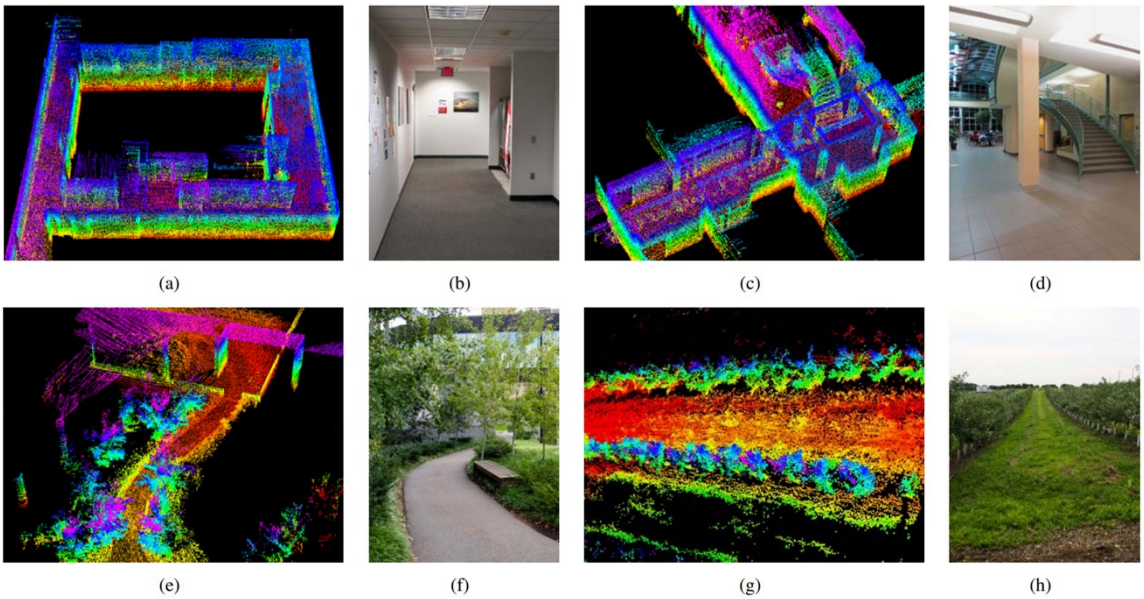


Figure 3.5: Maps generated in (a)-(b) a narrow and long corridor, (c)-(d) a large lobby, (e)-(f) a vegetated road, and (g)-(h) an orchard between two rows of trees [2].

LOAM is currently the most influential LiDAR SLAM algorithm as numerous advanced variants of it have been developed through the last years. The most celebrated ones are A-LOAM [118] and F-LOAM [119] which aim to simplify code structure and decrease computational cost, and LeGo-LOAM [120] that is a lightweight ground-optimized version of *LOAM*. However many more show great perspectives by demonstrating very promising results such as NDT-LOAM [121], and SA-LOAM [122]. Even more recent algorithms, namely LIO-mapping [123] and LIO-SAM [124], through the use of an IMU sensor, they enhance the traditional systems by implementing self-motion information which remarkably elevates their robustness and accuracy even in challenging conditions.

3.3 Visual SLAM

The immoderate need for autonomy in modern applications coupled with the excessive amount and low cost of camera sensors has led to a wave of different pioneering visual SLAM algorithms which try to make the breakthrough in the field. Having begun from probabilistic and then direct methods, visual SLAM problem is now mostly solved through feature-based methods and their modern variations. To adduce an example, although point features is the traditional and the most dominant option in the field, recent methods use different geometric features like line segments, or even high-level features such as semantically labeled objects. A summary table (Table 3.1) of the most representative visual SLAM systems and their related techniques for data association is presented below. In this section we will focus on the systems that inspired and have common characteristics to our work.

3.3.1 ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM

ORB-SLAM3 is currently the most accurate state-of-the-art visual SLAM library. The software includes many different modes, as it is available for pure visual or visual-inertial SLAM for monocular, stereo, RGB-D and fisheye cameras. The structure of the system is quite elaborate as the software has been developed for many years, starting from ORB-SLAM [125], and includes many different features ranging from integrating an IMU to the original approach which is purely visual, to incorporating a loop closing and map-merging module. In this subsection, we will not delve into the mathematical details of the algorithm which are very similar to the ones of our own system that will be thoroughly explained in the next chapters, and instead we are going to describe the different components of the system. The system's architecture is depicted in Figure 3.6.

Table 3.1: Summary of the most representative visual SLAM and odometry systems.

System	**SLAM or VO**	**Feature Detection**	**Data Association**	**Camera Model**
Mono-SLAM [18, 19, 20]	SLAM	Shi Tomasi	Correlation	Monocular
PTAM [21, 22, 23, 24]	SLAM	FAST	Pyramid SSD	Monocular
LSD-SLAM [25, 26, 27]	SLAM	Edgelets	Direct	Monocular, Stereo
SVO [28, 29, 30]	VO	FAST+Hi grad.	Direct	Monocular, Stereo, Fisheye
ORB-SLAM2 [31, 32]	SLAM	ORB	Descriptor	Monocular, Stereo
DSO [33, 34, 35, 36]	VO	High grad.	Direct	Monocular, Stereo, Fisheye
DSM [37, 38]	SLAM	High grad.	Direct	Monocular
PL-SLAM [13, 39]	SLAM	ORB+LSD	Descriptor	Monocular
PL-SLAM [3, 40]	SLAM	ORB+LSD	Descriptor	Stereo
PL-VIO [14, 41]	SLAM	ORB+LSD	Descriptor	Monocular + IMU
PL-VINS [42, 43]	SLAM	ORB+LSD	Descriptor	Monocular + IMU
MSCKF [44, 45, 46, 47]	VO	Shi Tomasi	Cross Correlation	Monocular, Stereo + IMU
OKVIS [48, 49, 50]	VO	BRISK	Descriptor	Monocular, Stereo, Fisheye + IMU
ROVIO [51, 52, 53]	VO	Shi Tomasi	Direct	Monocular, Stereo, Fisheye + IMU
ORBSLAM-VI [54]	SLAM	ORB	Descriptor	Monocular + IMU
VINS-Fusion [55, 56, 57]	VO	Shi Tomasi	KLT	Monocular, Stereo, Fisheye + IMU
VI-DSO [58]	VO	High grad.	Direct	Monocular + IMU
BASALT [59, 60]	VO	FAST	KLT (LSSD)	Stereo, Fisheye + IMU
Kimera [61, 62]	VO	Shi Tomasi	KLT	Stereo + IMU
ORB-SLAM3 [1, 63]	SLAM	ORB	Descriptor	Monocular, Stereo, Fisheye + IMU
ORB-LINE-SLAM [64] (ours)	SLAM	ORB+LSD or EDLines	Descriptor	Stereo + IMU

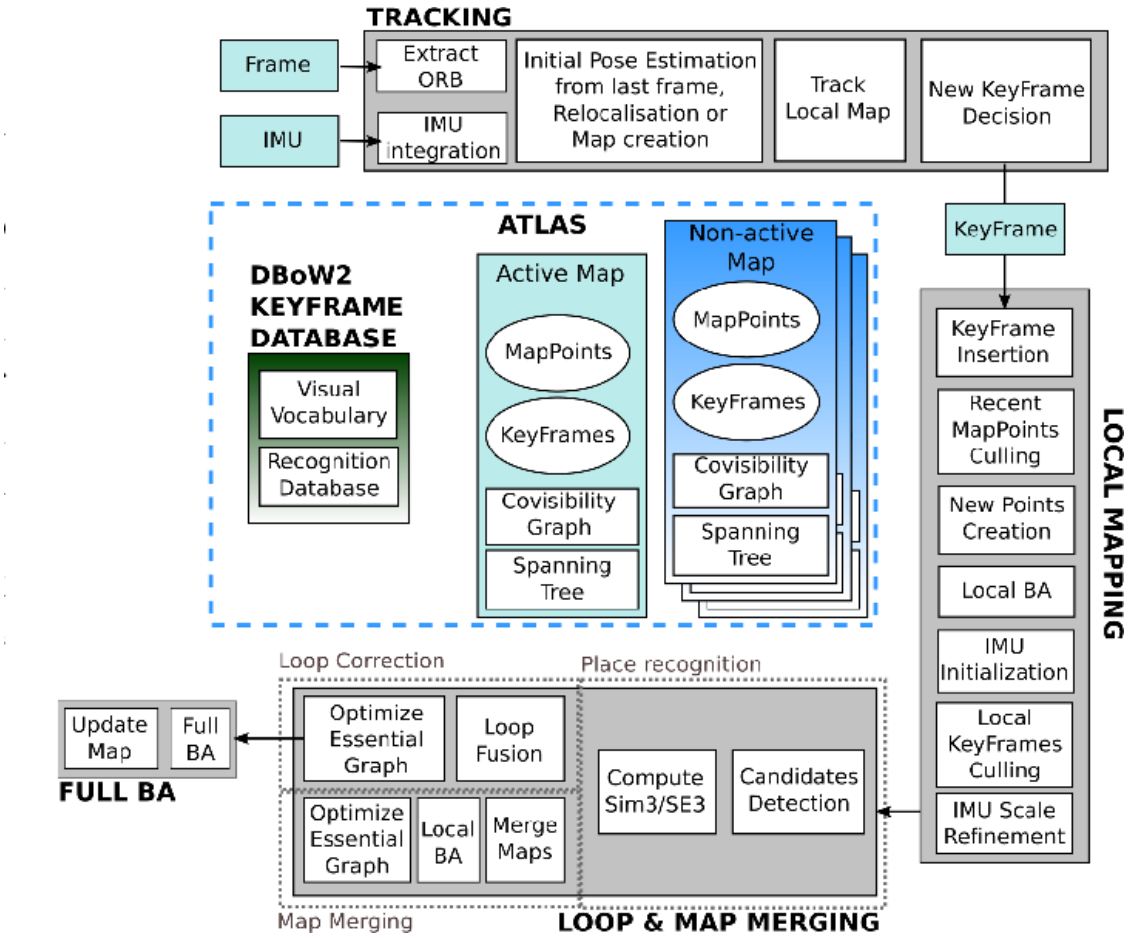


Figure 3.6: Structure of ORB-SLAM3 [1].

The system consists of 3 parallel threads that work together to secure the high performance of the algorithm. The first thread is the *tracking* thread which is the core part of the system. Fundamentally, it is responsible for calculating the main estimation for the pose of the camera and can be described as the visual odometry part of the algorithm. Throughout this thread the pose of the camera is optimized twice, one from finding correspondences between the current and the previous frame, and then from finding correspondences between the current frame and a local map. Regarding the visual-inertial part of the system, the state vector that needs to be estimated, not only includes the pose of the camera, but also the velocity in the world frame, and the gyroscope and accelerometer biases. These parameters are embedded in the system as the algorithm preintegrates the IMU measurements between consecutive visual frames by following the strategies from [126, 127]. Finally, the tracking thread adopts a policy for close and far points to decide if each frame is a keyframe and so the local mapping thread should be activated.

The next thread is the *local mapping* thread whose main purpose is to optimize the pose of the camera as well as the points of the map in a local window. The first step of the algorithm is to perform a culling of the recent points of the map by examining if each point is observable by enough keyframes in the local map. Subsequently, the system tries to take advantage of previously untapped points for which the algorithm did not find a pair, by searching to match them with other unmatched points in keyframes connected in the covisibility graph (explained in the next section). In the next and most crucial step, the system performs a local bundle adjustment to optimize the pose of the camera and the position of the points in the map in a local window, alleviating this way partially the accumulated drift both in the estimation of the camera and the map. Lastly, a local culling of the keyframes is executed by inspecting the percentage of the redundant map points which already exist in other keyframes of the local map.

The last thread is the *loop closing* thread which is a key for the robustness and efficacy of ORB-SLAM3. Specifically, if the tracking is lost, instead of collapsing, the algorithm creates and a new map and stores the previous in a database called ATLAS. In every frame, the loop closing thread performs a place recognition so as to detect candidates for merging the maps. If the candidate is accepted, the map is merged and the map as well as the pose of the camera are first corrected through a pose-graph optimization and then again optimized through a local bundle adjustment. In this way, the algorithm manages to survive even when it can not detect enough correspondences, as long as it can revisit the same scene. The second functionality of this thread is the loop closing process. In a similar way as in the map merging, in every frame the loop closing thread tries to detect a possible pair to close a loop. Hence, by revisiting previously seen areas, the system manages to close loops and thus significantly alleviate the accumulated drift during the whole journey by performing a pose-graph optimization and a global Bundle Adjustment.

3.3.2 SLAM with Point and Line Features

In this subsection we discuss related work on point-line SLAM systems. One of the most influential ones in the field, which has also inspired our work, is PL-SLAM [3], the first real-time system that integrated point with line features for stereo cameras. Similarly as in most modern visual SLAM approaches, PL-SLAM is a keyframe-based system that optimizes the position of the camera as well as the features of the map through Bundle Adjustment. As before, the system can be divided into three separate parts: the stereo visual odometry, the local mapping, and the loop closing (Figure 3.7). In the stereo visual odometry, the matching algorithm compares the descriptors of the point and line features, and the process gets accepted only if the candidates are best mutual matches, a policy that we have also adopted for the line features in our work. Afterwards, similarly to the first optimization of the tracking thread of ORB-SLAM3, the system performs a frame-to-frame tracking and optimizes the pose of the camera by finding correspondences between the current and the previous frame. The algorithm that was chosen to solve

the nonlinear least-squares problem concerning Bundle Adjustment is the Gauss-Newton described in section 2.2.4. To incorporate the line segments into the optimization process, instead of minimizing the 2D distances between the endpoints of the corresponding line segments, the authors of PL-SLAM introduced an efficient error function that we have also used in our algorithm and thoroughly explain in Chapter 5. For selecting keyframes, this method adopts the idea that a keyframe (KF) should be inserted so that the camera is always close enough to one of them and thus no drift is accumulated. Specifically, by following the scheme of [65], it inserts a keyframe only if the entropy ratio between the motion estimate from the last keyframe to the current frame and the motion from the last keyframe to the very next frame, lies below the experimentally established threshold of 0.9. The local mapping part is quite similar to the one of ORB-SLAM3 as it firstly creates new features by adopting the same strategy and then performs a local Bundle Adjustment to optimize the map features and the pose of the camera in the local window. Lastly, one of the main contributions of this work is an extended bag-of-words approach that takes into account both points and line features and is extensively used in the place recognition process for the loop closing part. To detect a loop, the algorithm does not only match the frame with the best candidate, but also this frame must be similar to the surrounding keyframes of the best candidate. If a loop is accepted, again a pose-graph optimization and a global Bundle Adjustment are performed to optimize the estimated trajectory of the camera and the features of the map.

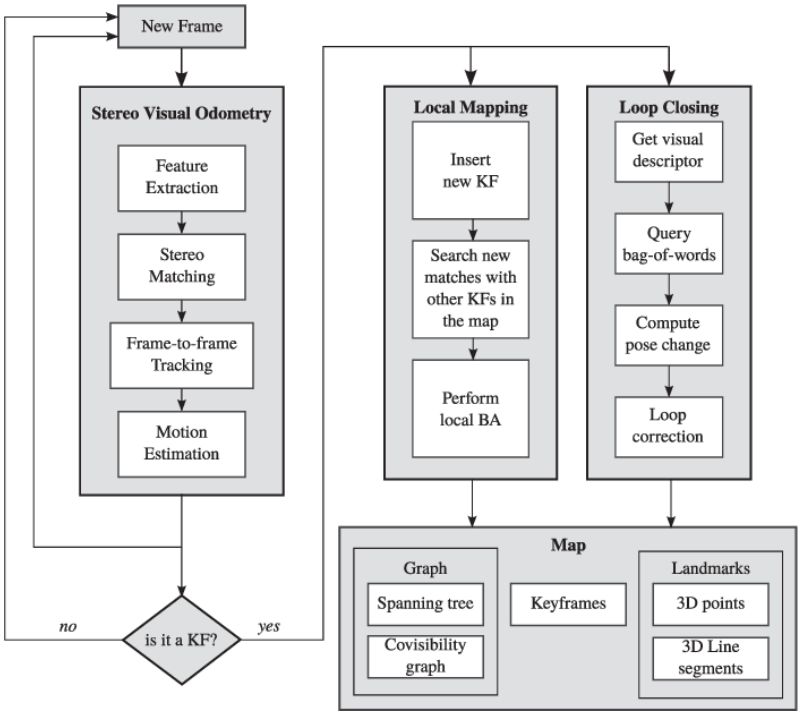


Figure 3.7: Structure of PL-SLAM [3].

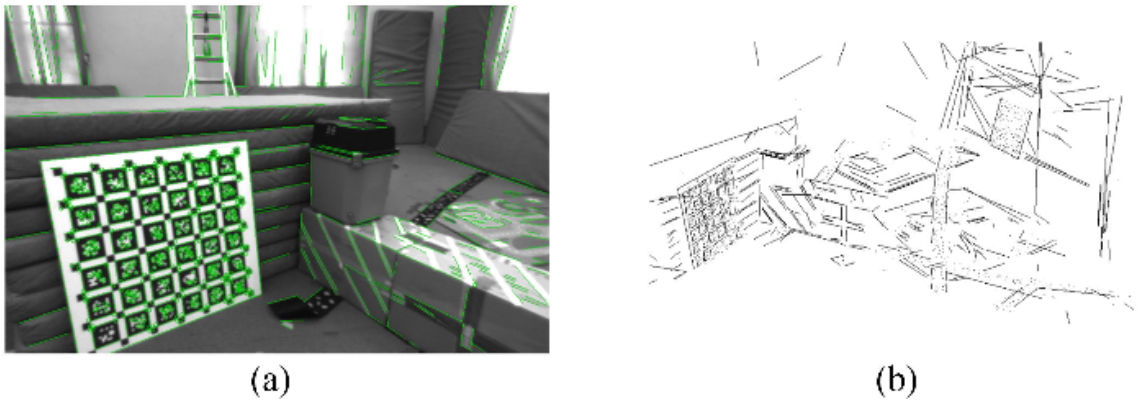


Figure 3.8: Map reconstruction by PL-SLAM [3].

A work of similar philosophy is monocular PL-SLAM [13], which embodies line features into the core of ORB-SLAM [125]. Similarly as in our method, the lines are included only in the tracking and local mapping process and not in the loop closing thread, as in this case the computational cost would considerably elevate and consequently affect negatively the real-time performance of the system. The error function used in the optimization process regarding the line segments is fairly the same as in stereo PL-SLAM [3], with just using two parameters in the error function for the line segments, one for each endpoint, instead of only one. An important contribution of this work is also the proposal of a novel initialization strategy for monocular cameras that operates exclusively with line segments. This technique relies on finding line correspondences between three consecutive frames with the assumption that the rotation between them remains constant. The accuracy of this system significantly outperformed ORB-SLAM especially in low-textured scenes, pointing in this way how importantly lines can assist point features to increase the robustness and efficacy of monocular SLAM (Figure 3.9).

Another noteworthy work that incorporates line segments is PL-VIO [14], which is a monocular point-line visual-inertial odometry system. As usual in the field, after this method tracks point and line features and pre-integrates the IMU measurement, it optimizes the pose of the camera by minimizing a cost function which combines both the pre-integrated IMU error term and the point and line re-projection error terms. An innovation of the algorithm is that space lines are described through Plücker coordinates and orthonormal representation so as to reinforce their computational simplicity and representational compactness. As for the performance of the system, it constitutes another example of how line segments can assist an algorithm to create more vivid representations of the environment (Figure 3.11) as well as bolster its efficacy. Specifically, according to [14], PL-VIO outperformed some of the most celebrated systems, particularly ROVIO [51], OKVIS-Mono [48], and VINS-Mono [55], in the EuRoC dataset [73] (Figure 3.10). However, a major drawback of the method is its high computational cost which is a bottleneck for its real-time performance. In order to overcome this impediment, in a way

similar as we have implemented in our work, PL-VINS [42] uses a custom LSD [66] for the line detection and additionally filters out lines according to their geometric attributes, significantly reducing this way the computational burden.

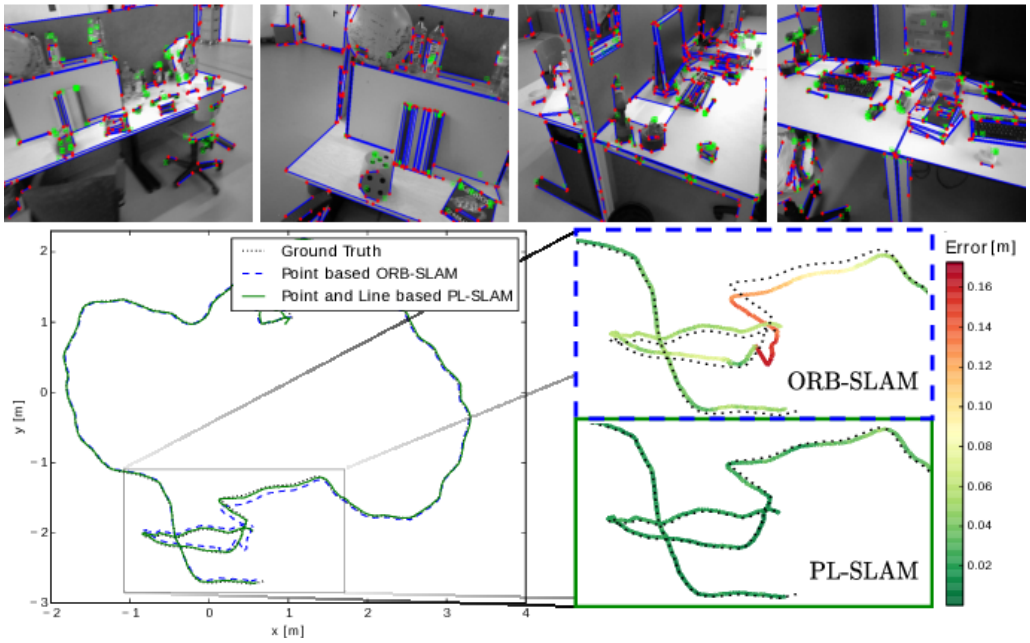
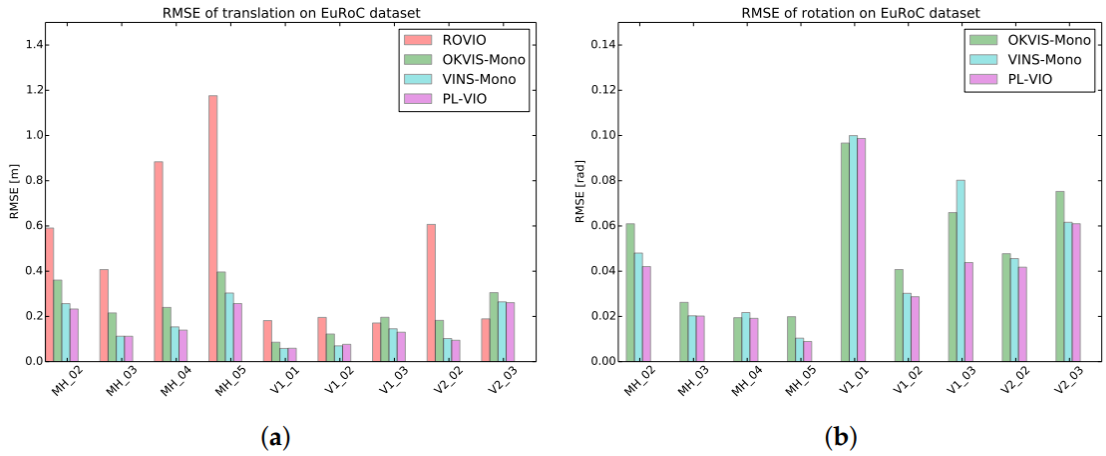


Figure 3.9: **Top**: Point and Line detection with PL-SLAM. **Bottom**: Performance comparison of ORB-SLAM and PL-SLAM in TUM dataset [12] [13].



(a) (b)

Figure 3.10: Performance comparison of PL-VIO with other state-of-the-art monocular visual–inertial methods in EuRoC dataset [14].

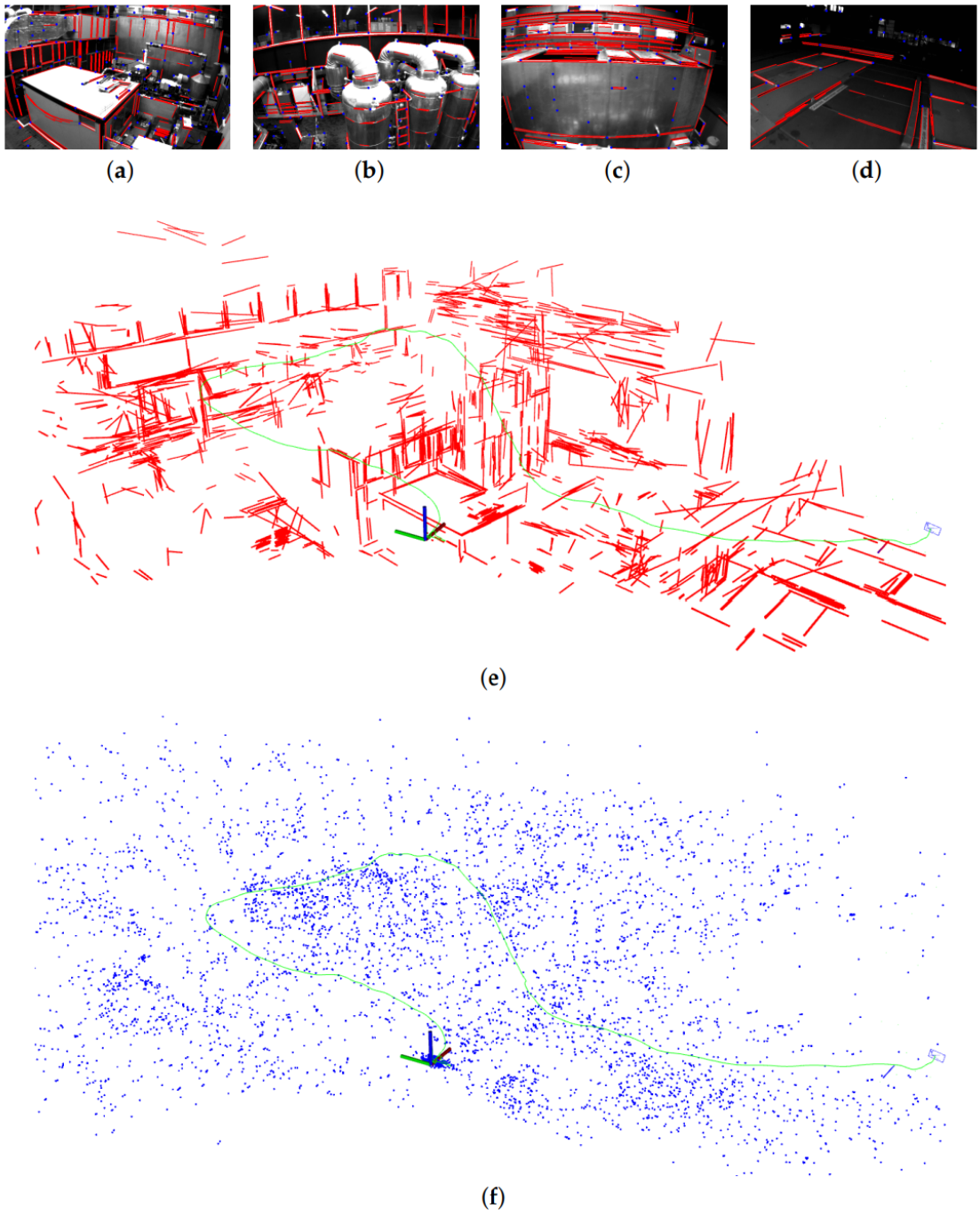


Figure 3.11: Mapping results in the MH05 difficult sequence of EuRoC dataset. (a–d) Detected point and line features for different frames. (e) The reconstructed line map (red) and the trajectory (green). The scenes with rich line features (such as the window, baluster, and cabinet) are clearly reconstructed from the map. (f) The reconstructed point map (blue). As compared to the line map, the structure is hard to identify in the point map [14].

Chapter 4

Overview of ORB-LINE-SLAM System

4.1 Introduction

The central idea behind our work was to build a hybrid system, whose main core would be a point-line algorithm in which the line features would have an assistive role to the point features so as to elevate the robustness and efficacy of the original system in demanding conditions. Concurrently, we had the motivation to develop a novel only-line feature-based algorithm that would be efficient enough to be independent of the use of point features. Such a hybrid system with standalone nodes could eventually be used much more effectively in the field of visual SLAM. Thus, the system contains two different modes, and the user can choose between the point-line or the only-line SLAM. Both algorithms are built upon the code and the philosophy of ORB-SLAM3[1], which is the successor of ORB-SLAM-ATLAS[128], ORB-SLAM-Visual-Inertial[54], ORB-SLAM2[31], and ORB-SLAM[125].

The general structure of the system, as extended to include line segments, is depicted in Figure 4.1 and its main components are summarized below for a smooth introduction to the reader. The system is divided into three separate parallel threads. With sequential order, the first one is the tracking thread, in which an initial pose estimation is calculated either by minimizing the re-projection error functions of the map point and line matches detected between the current and previous frame, or if the tracking is lost by creating a new map. Subsequently, the pose is optimized again within the tracking thread by finding the point and line feature matches between the current frame and a local map and by performing again a motion-only bundle adjustment just as in the initial pose estimation. The second thread concerns local mapping, in which both the position of the point and line features as well as the pose of a cluster of keyframes are optimized by performing a local bundle adjustment. Finally, the last thread is the loop closing, in which loops and map merges are detected through the use of a Place Recognition module based on DBoW2 [129], followed by a pose-graph optimization performed so as to diminish the accumulated drift. In the case of map merging, a local bundle adjustment is performed before the pose-graph optimization, whereas in the case of a loop closing, after the pose-graph optimization a fourth thread launches and independently performs a global bundle adjustment that optimizes the whole map and the pose of the associated keyframes. The loop closing thread operates exclusively with the use of points, and the line features are included so that just their position is corrected after closing a loop or merging a map according to an aligning transformation and a pose-graph optimization, in order to be accurately used on the rest of the sequence. Hence the structure of the loop closing thread stays exactly the same as in ORB-SLAM3. The system also includes a covisibility graph [130] which connects the keyframes that share same points and lines and a minimum spanning tree that links all the keyframes together. Its functionality is to allow the tracking and the local mapping threads to operate locally by using each time only a cluster of keyframes, while also playing a major role in the loop closing and map merging processes, used as structure in the pose-graph optimization.

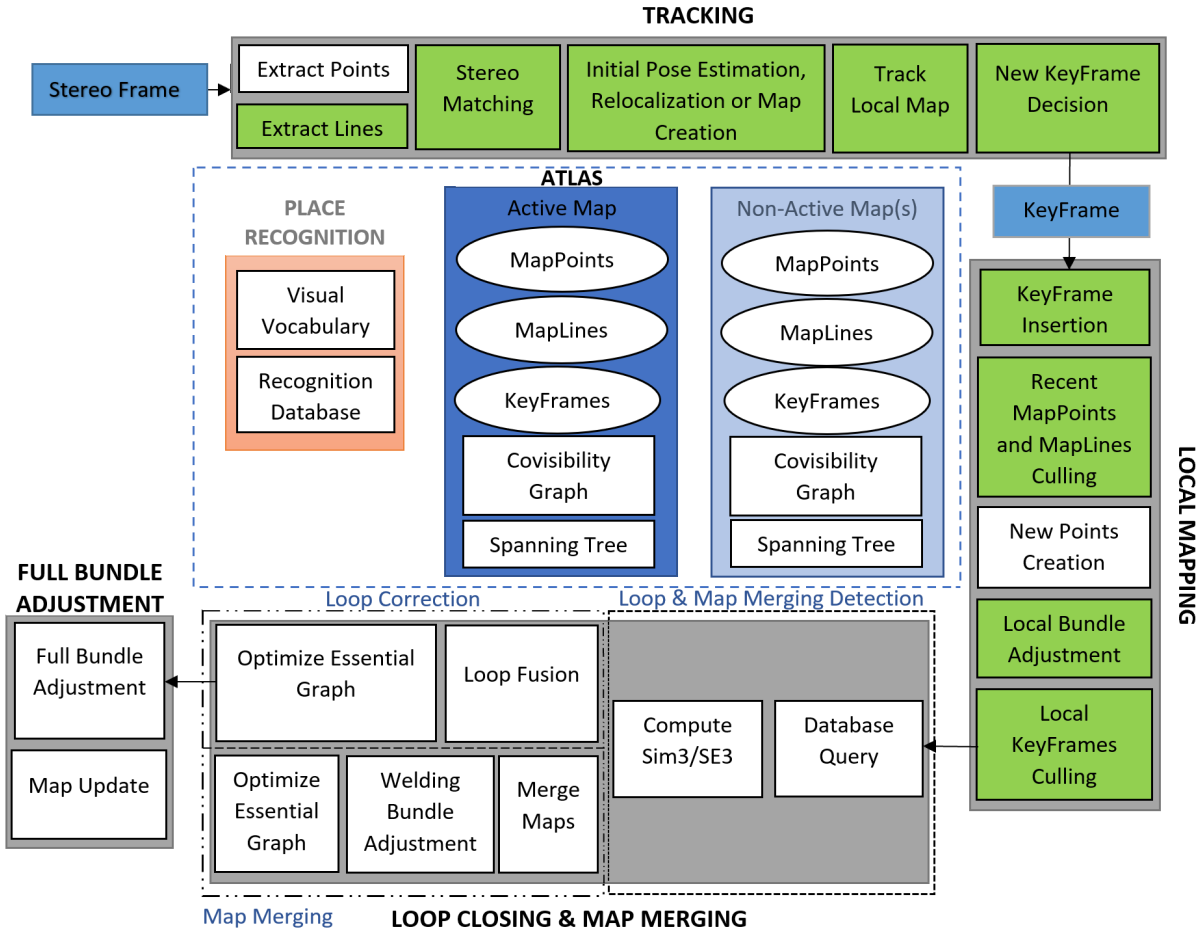


Figure 4.1: Overview of the general structure of the system. The components filled with green color are the ones in which the line segments are included.

4.2 Tracking Thread

4.2.1 Feature Selection and Line Matching

Regarding the line feature selection, the system includes both the Line Segment Detector (LSD) method[66] and the EDlines[67] and the user can choose between them. In particular, we used the custom LSD algorithm implemented in the OpenCV library[68] and the original implementation of EDlines found on github unmodified. Our primary purpose was to experiment with both, so as to compare them and decide which one is the most suitable for the visual SLAM process. According to our experiments, the LSD algorithm is considerably more effective than the EDlines, as it shows significantly better repeatability in detecting the same line features, and hence much more line matches can

be determined. For this reason, all our experiments in the evaluation chapter (chapter 6) were made by using the LSD algorithm. In fact, with the use of EDlines, the only-line algorithm fails to finish almost all the sequences of the EuRoC dataset, while even in the ones that it manages to finish the results achieved are very poor (specifically larger than 1.5m RMSE ATE). In terms of computing time, the EDlines algorithm is much more efficient, as the time required for the detection of the line segments is approximately two to six times faster depending on the scene in comparison to the LSD method, and from this scope EDlines might be a more appropriate choice. However, the selected custom LSD method gives the ability to the user to tune the scale of the image in which the line segments will be detected, reducing significantly the computing time. Even with an initial scale of 0.5 the LSD method is much more effective than the EDlines while it requires almost the same computational time. Another key factor to the efficiency of our method and the ability of the only-line algorithm to operate effectively is that the line segments are extracted from 2 scale levels with a scale factor equal to 2 and therefore their number is sufficient for the algorithm to determine enough line matches. Regarding the point features, the system uses the code of ORB-SLAM3 and thus the ORB[83] method is used as it provides a plethora of advantages for the visual SLAM process.

For the line matching process, to find correspondences between the line segments, we selected the 256 bits Line Band Descriptor (LBD)[69] which represents each line segment according to its local appearance. Regarding the stereo matching, to expedite the procedure, we create a grid structure and, for each line segment of the left image, we search only in a fixed window of this grid structure in the right image close to the position of the line that we are searching to match. In our case that we use rectified images, the size of the window remains relatively small depending also to the resolution of the images, so as to decrease the computational burden. However, for non-rectified images it is suggested to use a larger window as the corresponding line in the right image is maybe displaced relatively to the position of the line in the left image. This concept is explained in detail in subsection 2.2.2. Subsequently, identically to PL-SLAM[3], in order to reduce the outliers, we compare the distances between the descriptor of the line segment of the left image and its candidates and we determine the one with the smaller distance. Then, we repeat this procedure vice versa and if the two best candidates concur then a matching set of line segments is determined. However, in the intermediate time we also do an extra filtering concerning the direction of the corresponding line segments. Particularly, we examine if their cosine similarity, which expresses the cosine of the angle between the two line segments and determines whether they are pointing in roughly the same direction, exceeds the threshold of 0.75 radians which equals approximately to 41 degrees. Mathematically, this relation can be expressed by the equation:

$$\text{similarity}(x, y) = \cos(\theta) = \frac{x \cdot y}{|x||y|} < 0.75. \tag{4.1}$$

As for the line matches used for the prediction of the pose and the tracking of the local

map, we adopt the KNN method using the hamming distance; again, both lines have to be mutually best candidates for the matching process to be successful.

4.2.2 Motion Estimation

Just before the estimation of the pose of the camera, the system needs to ensure that there are enough inliers so as to accurately determine it. For this reason, in the initialization frame, i.e. the frame in which the algorithm starts the tracking process, at least 500 ORBS and line segments need to be detected in total. If the algorithm does not manage to detect this number of features, the tracking thread stops and seeks to detect sufficient nomber of features in the next frame. Afterwards, as already mentioned, motion estimation is conducted by first making an initial pose prediction between the current and the previous frame through a motion-only bundle adjustment. Like in the stereo line matching, the system searches for correspondences between the two frames by using a window for each feature. If the system can not find enough correspondences, then it uses a wider search of the map points around their position in the last frame. If the procedure fails again, the system assumes that the tracking is lost and tries to do a global relocalization by searching to match the frame with a keyframe in the database through the place recognition module which essentially interprets the frames into bags of words. The next step of the method is to track the local map, whose structure is going to be explained later in this subsection, by performing again a motion-only bundle adjustment. In both procedures, by taking advantage of the rich geometrical information that the line features provide, we filter out those line matches with significantly different orientations and positions in the image. Specifically, after the re-projection of the map line segments on the current frame, the conditions that must be satisfied by the set of lines are:

1. The orientations of the two lines must not differ more than $\pi/8$.

2. The horizontal and vertical pixel distance between the starting points and between the ending points of the two line segments, must not exceed the pixel width and height of the image divided by 10, respectively.

The importance of this filtering relates primarily to the fact that, during the bundle adjustment, the re-projection error function includes the distance between the endpoints of the line segment and its corresponding infinite line, thus resulting to situations where some obvious mismatches may not be considered as outliers during the optimization process. To adduce an example, if the two corresponding line segments belong to the same infinite line or to two infinite lines that almost concur, then the associated error would be precisely or close to zero. However, these line segments may be completely displaced inside the image and coincidentally happened to have common descriptors and the one to concur with the infinite line of the other. With this filtering, not only we

assure that such mismatches will be excluded because of the second condition, but also that obvious mismatches of lines that have significantly different directions will be filtered too.

The local map that is used for tracking comprises three different sets of keyframes. The first one Λ_1 shares map points and lines with the current frame, while the second one Λ_2 contains neighbors of the keyframes of Λ_1 in the covisibility graph. The third set Λ_3 includes a reference keyframe $\Lambda_{ref} \in \Lambda_1$ which shares the largest possible set of map points and lines with the current frame. The policy of matching map points seen in Λ_1 and Λ_2 with points in the current frame remains the same as in [125]. For the corresponding process concerning line segments, we have adopted the following strategy:

1. After re-projecting the map line segment to the current frame, we check if at least one of its endpoints lays out of the image bounds and if true the line is discarded.

2. We calculate the angle between the current viewing ray $\mathbf{v}$ and the mean of the viewing direction $\mathbf{n}$ of the midpoint of the map line segment. If $\mathbf{v} \cdot \mathbf{n} < \cos(60°)$ then the line segment is discarded.

3. We calculate the distance d from the midpoint of the map line segment to the camera center and discard the line if d is out of the scale invariance region of the midpoint, i.e. $d \notin [d_{min}, d_{max}]$.

4. We compare the descriptor of each map line with the descriptors of the unmatched lines in the current frame, and we associate two line features only if they are best mutual candidates.

The motion-only bundle adjustment which follows the aforementioned process is thoroughly explained in chapter 5.

4.2.3 Keyframe Selection

Our system follows a policy similar to ORB-SLAM2 [31] for the keyframe selection. In particular, it uses the same conditions as ORB-SLAM and it also includes an extra modified condition regarding close and far points and lines similar to ORB-SLAM2. For the classification of close and far keypoints and keylines a threshold of ~ 40 times the stereo baseline, as suggested in [70], is applied. If the associated depth of a keypoint is less than this threshold, then it is classified as close, otherwise it is classified as far. Similarly, if the associated depth of both the endpoints of a keyline is less than this threshold then it is classified as close, otherwise it is classified as far. Close keypoints and keylines provide rich information about translation, and rotation, while far ones provide accurate information about rotation but weaker about translation. In challenging environments in which the scene is mostly placed far from the stereo sensor, it is imperative to have a sufficient

number of close keypoints and keylines so as to accurately estimate the translation of the camera. Hence, two thresholds were empirically established. In the case of the point-line algorithm, if the number of tracked close points and lines drops below a value (set to 140) and the frame creates more than a fixed number of new close stereo points and lines (set to 100), the system will insert a new keyframe. Similarly, regarding the only-line algorithm, a new keyframe is inserted if the number of tracked close lines drops below a value (empirically set to 40) and the frame creates more than a set number of new close lines (in our case, 30).

4.3 Local Mapping

4.3.1 Recent Map Points and Lines Culling

After the insertion of a keyframe, the system follows a restrictive policy for culling recent map points and lines. By culling the redundant map features the system ensures that the map does not contain non trackable features that emerge from wrong triangulations due to possible mismatches, and concurrently that the map is much more meaningful as it does not contain meaningless features which adulterate the reconstructed scene. The conditions that the points must fulfill remain the same as in ORB-SLAM[125], while for the lines the conditions applied are the following:

1. The line must be found in more than 20% of the frames in which it is visible through their local map.

2. If more than one keyframe has passed from the creation of the line, it must be observed from at least two keyframes.

Subsequently, the system creates new map points by taking advantage of points that were previously unmatched. Particularly, the system tries to find correspondences for the unmatched points in keyframes connected to the covisibility graph. So that a match gets accepted and the method triangulates the point to the map as explained in section 2.2.3, first the match must fulfill an epipolar constraint, and also positive depth, scale consistency, and reasonable parallax and reprojection error need to be guaranteed. This procedure was difficult to implement by using the line features, as their endpoints are considerably unstable especially when observed from fundamentally different viewpoints, meaning that accurate triangulation is most of the times not feasible. For this reason, the lines are not included at this stage of the system, requiring further research and effort to incorporate them properly. The process of local bundle adjustment which follows is thoroughly described in chapter 5.

4.3.2 Keyframe Culling

The process of keyframe culling allows the system to discard redundant keyframes that do not contain any significant additional information with respect to the ones that already exist, thus also reducing the computational time required for the local and global bundle adjustment as fewer keyframes are included in the process. In particular, a keyframe is considered redundant if at least 90% of its map points and lines (in the case of the only-line algorithm, of its map lines alone) have been seen in at least three different keyframes in the same or better scale. In this way, the algorithm avoids to insert keyframes that depict exactly the same scene information, thus eventually creating a much more meaningful, compact and coherent reconstruction, while simultaneously it ensures that the keyframes that survived this filtering are the ones to contain the most accurate information due to the scale condition.

4.4 Loop Closing and Map Merging

The first step of the *Loop Closing* and *Map Merging* processes is the detection of loop candidates. To begin with, the method through the bag of words DBoW2 [129] seeks to find the 3 most similar keyframes K_m to the active keyframe K_a excluding keyframes covisible to K_a. To achieve this, it compares the descriptors of the points inside a window of the current image with the ones of the map points and tries to find associations. To avoid mismatches, it also checks if the distance of the second-best correspondence candidate is close to the one of the first. Particularly, for each keyframe k_m the algorithm sets a local window which includes its best covisible keyframes and their common map points. Through DBoW2 putative correspondences between the points in this local window and keypoints from k_a are determined. Then, using RANSAC algorithm [131] the system computes the transformation T_{am} that best aligns the map points of the local window with the ones in K_a. After applying the transformation, if the reprojection error of a putative match is below an established threshold, then this pair gives a positive vote for this candidate. The candidate with the most votes, if these exceed a fixed minimum, will be selected to close a loop. The next goal of the algorithm is the refinement of the transformation T_{am}. First, all the map points in the local window are transformed through T_{am} and the system seeks to find correspondences between them and the keypoints in k_a. Then, a reverse matching is performed by trying to associate keypoints from k_a with points in the keyframes of the local window. Having determined the correspondences both ways, the transformation T_{am} is refined through a non-linear optimization by using the bidirectional reprojection error from all the matches. This procedure is repeated and the transformation T_{am} is refined again, this time by using a smaller image search window, only if the inliers after the first optimization, i.e., the matches that got accepted during the optimization, exceed a predetermined threshold.

The last step of the loop detection process is the verification of the loop. Specifically, the algorithm searches in the active part of the map to find enough correspondences between the points of two keyframes covisible with K_a and the points in the local window. If the associations surpass a fixed value, then this keyframe casts a positive vote. The process is terminated, either if 3 keyframes verify the loop, or if two consecutive keyframes reject it.

When the system detects a loop between the keyframes K_a which belongs to the active map M_a, and k_m which belongs to a previous map M_m stored in Atlas [128], it merges the two maps. The two maps are merged by aligning the active map M_a to M_m through the aligning transformation T_{ma}. Since this process can be very time-consuming, the map merging is divided into two separate steps. In the first step, the algorithm uses a welding window that includes k_a, k_m, their covisible keyframes, and all the map points observed by them, and aligns the keyframes and the associated points of M_a to M_m through the previously mentioned aligning transformation. As soon as the two maps are fused, the algorithm purges duplications of M_a points in M_m map to refine the new active map. Next, a local bundle adjustment is performed to optimize all the keyframes and map points associated to the welding window. The keyframes of M_m that are not members but observe points of the welding window, participate in the local bundle adjustment but with their poses fixed to eliminate the gauge freedom. The last step of the map merging procedure is a pose-graph optimization that rectifies the rest of the map by keeping fixed the welding window.

The loop closing process is very similar to map merging, with the distinction that both keyframes belong to the same map and that the pose-graph optimization precedes the bundle adjustment. As before, a welding window from the corresponding keyframes is formed and accordingly the loop is corrected through an aligning transformation. Subsequently, duplicated points are located and fused refreshing this way the links in the covisibility and essential graphs. In contradiction with map merging, the next step is the pose-graph optimization which spreads the correction throughout the map. A global bundle adjustment is performed finally to re-optimize the map and the estimated trajectory of the camera. The significance of loop closing is of pivotal importance if someone thinks that this is the only means of the algorithm to correct the drift which is constantly integrating. Without its presence, after a time frame no pure visual localization algorithm could provide accurate results about the pose of the camera as the error would have accumulated so much to the estimation that the results would be completely erroneous. An example of how the estimated trajectory of a robot is corrected through loop closing is depicted in Figure 4.2.

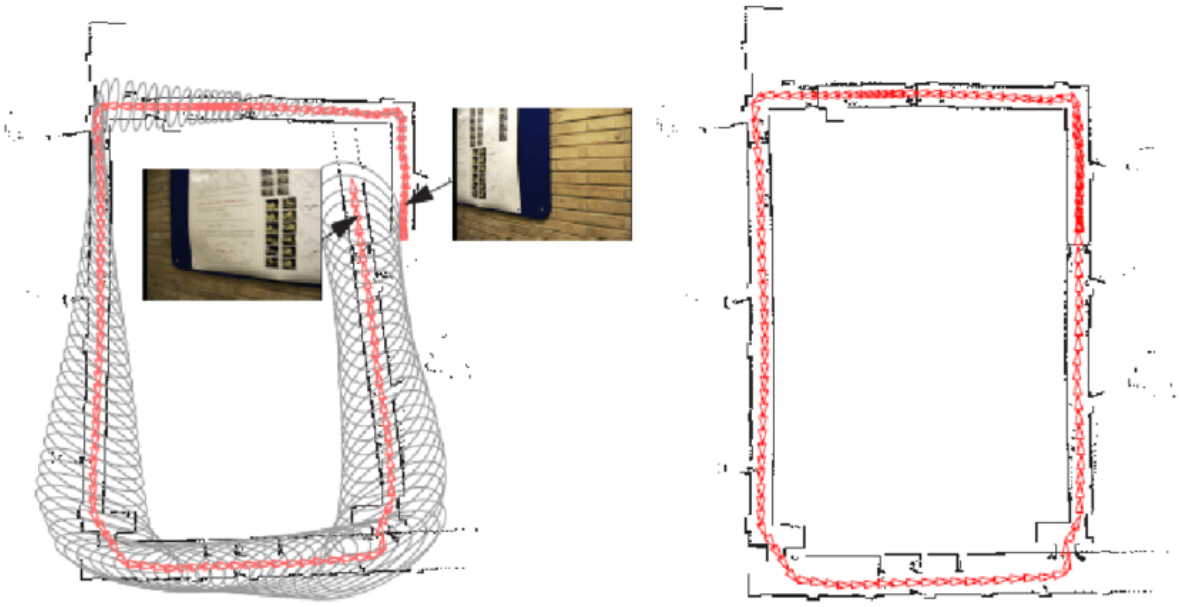


Figure 4.2: A snapshot of the estimated trajectory of a SLAM system before closing a loop is presented on the left. The corrected estimated trajectory after closing a loop is presented on the right [15].

Chapter 5

Bundle Adjustment

In this chapter we expound various error functions for the line segments and also the mathematical formation of the different types of bundle adjustment. Finishing, there is assiduous discussion and experimental analysis about the development of the adapting factor.

5.1 Error Function Selection for the Line Segments

In order to choose the most efficient error function for the line segments, we adopted in our system and compared the most widely used ones that are found in the literature. The conventional error function that is used for the point features is the 2D distance between the re-projected 3D point and its corresponding point in the image. However, this procedure cannot be directly used for the line segments, as their endpoints are unstable and therefore displaced or even occluded from one frame to the next one.

Let $\Xi_{iw} \in \mathrm{SE}(3)$ denote the camera pose of the i-th frame, $P_{w,k}$, $Q_{w,k} \in \mathbb{R}^3$ be the 3D endpoints of the k-th line segment observed by the i-th frame and $p_{i,k}, q_{i,k} \in \mathbb{R}^2$ be the 2D endpoints in the image plane that correspond to the endpoints of the k-th line segment in the i-th frame, where w stands for the world reference. The three error functions adopted in our system are defined as follows:

5.1.1 Euclidean distance

The first error function, as presented in [3], uses the Euclidean distances between the endpoints of the re-projected 3D line segment and its corresponding infinite line in the image plane. The equation of the infinite line can be obtained through the cross product between $p_{i,k}$ and $q_{i,k}$ and is explicitly defined as:

$$l_{i,k} = \frac{{}^{h}p_{i,k} \times {}^{h}q_{i,k}}{\|{}^{h}p_{i,k} \times {}^{h}q_{i,k}\|}, \tag{5.1}$$

where ${}^{h}p_{i,k}, {}^{h}p_{i,k} \in \mathbb{R}^3$ stands for the homogeneous coordinates of $p_{i,k}, q_{i,k}$. Hence the error function can be formulated as:

$$e_{i,k} = \begin{bmatrix} e_{i,k}(1) \\ e_{i,k}(2) \end{bmatrix} = \begin{bmatrix} l_{i,k} \cdot {}^{h}\pi(\Xi_{iw}, P_{w,k}) \\ l_{i,k} \cdot {}^{h}\pi(\Xi_{iw}, Q_{w,k}) \end{bmatrix}, \tag{5.2}$$

where ${}^{h}\pi(\Xi_{iw}, P_{w,k})$, ${}^{h}\pi(\Xi_{iw}, Q_{w,k})$ stands for the homogeneous coordinates of $\pi(\Xi_{iw}, P_{w,k})$, $\pi(\Xi_{iw}, Q_{w,k})$, which represent the 2D endpoints of the re-projected line segment in the image plane and can be defined as:

$$\pi(\Xi_{iw}, P_{w,k}) = \begin{bmatrix} f_x \dfrac{m_x}{m_z} + c_x \\ f_y \dfrac{m_y}{m_z} + c_y \end{bmatrix}, \tag{5.3}$$

$$\pi(\Xi_{iw}, Q_{w,k}) = \begin{bmatrix} f_x \dfrac{g_x}{g_z} + c_x \\ f_y \dfrac{g_y}{g_z} + c_y \end{bmatrix}, \tag{5.4}$$

$$m = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T = R_{iw} P_{w,k} + t_{iw}, \tag{5.5}$$

$$g = \begin{bmatrix} g_x & g_y & g_z \end{bmatrix}^T = R_{iw} Q_{w,k} + t_{iw}, \tag{5.6}$$

where $R_{iw} \in \mathrm{SO}(3), t_{iw} \in \mathbb{R}^3$ are the rotation and translation matrices related to Ξ_{iw} respectively, (f_x, f_y) is the focal length, and (c_x, c_y) is the principal point, both known from the calibration of the camera. To optimize the pose of the camera by using the Levenberg–Marquardt method, first the Jacobian matrix needs to be derived as explained in 2.2.4 and 2.2.5. The Jacobian matrix can easily be computed from the equations 5.1-5.6 via the chain rule. Explicitly it can be described through the equation:

$$J_1 = \frac{\partial e_{i,k}}{\partial \Xi_{iw}} = \begin{bmatrix} \begin{bmatrix} l_0 \\ l_1 \end{bmatrix}^T \dfrac{\partial \pi(\Xi_{iw}, P_{w,k})}{\partial m} \dfrac{\partial m}{\partial \Xi_{iw}} \\ \begin{bmatrix} l_0 \\ l_1 \end{bmatrix}^T \dfrac{\partial \pi(\Xi_{iw}, Q_{w,k})}{\partial g} \dfrac{\partial g}{\partial \Xi_{iw}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{f_x}{m_z} l_0 & \frac{f_y}{m_z} l_1 & -\frac{f_x m_x l_0 + f_y m_y l_1}{m_z^2} & -f_x \frac{m_x m_y}{m_z^2} l_0 - f_y (1 + \frac{m_y^2}{m_z^2}) l_1 & f_x (1 + \frac{m_x^2}{m_z^2}) l_0 + f_y \frac{m_x m_y}{m_z^2} l_1 & -f_x \frac{m_y}{m_z} l_0 + f_y \frac{m_x}{m_z} l_1 \\ \frac{f_x}{g_z} l_0 & \frac{f_y}{g_z} l_1 & -\frac{f_x g_x l_0 + f_y g_y l_1}{g_z^2} & -f_x \frac{g_x g_y}{g_z^2} l_0 - f_y (1 + \frac{g_y^2}{g_z^2}) l_1 & f_x (1 + \frac{g_x^2}{g_z^2}) l_0 + f_y \frac{g_x g_y}{g_z^2} l_1 & -f_x \frac{g_y}{g_z} l_0 + f_y \frac{g_x}{g_z} l_1 \end{bmatrix}. \tag{5.7}$$

During the bundle adjustment, a set of matched lines is filtered as an outlier only if $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 7.8$, where $\Sigma_{e_{i,k}}$ is the covariance matrix which is set to identity. Thus, this inequality can be simplified as $e_{i,k}^2(1) + e_{i,k}^2(2) > 7.8$, where $e_{i,k}(1)$ and $e_{i,k}(2)$ are the euclidean distances expressed in pixels depicted in Figure 5.1. So, this threshold conveys that if for instance $e_{i,k}(1)$ and $e_{i,k}(2)$ are equal to 2 pixels each, this pair will be excluded from the optimization process. As usual in the field, this threshold was adjusted empirically, and it worked well in all of our experiments.

5.1.2 Angle distance

The second error function, as introduced in [71], uses the angle distance between the observed and the re-projected line. Assuming the same definitions that we previously made and if $p'_{i,k}, q'_{i,k} \in \mathbb{R}^2$ represent the 2D endpoints of the re-projected line segment in the image plane just as $\pi(\Xi_{iw}, P_{w,k}), \pi(\Xi_{iw}, Q_{w,k})$, then the error function is formulated

as:

$$e_{i,k} = \begin{bmatrix} e_{i,k}(3) \\ e_{i,k}(4) \end{bmatrix} = \begin{bmatrix} \dfrac{q_{i,k}\vec{p'}_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|q_{i,k}\vec{p'}_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} - 1 \\ \dfrac{p_{i,k}\vec{q'}_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|p_{i,k}\vec{q'}_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} + 1 \end{bmatrix}. \tag{5.8}$$

The error function represents the cosine angle between the vectors $q_{i,k}\vec{p'}_{i,k}, q_{i,k}\vec{p}_{i,k}$ and $p_{i,k}\vec{q'}_{i,k}, q_{i,k}\vec{p}_{i,k}$. Like before, to apply the non-linear optimization method we need to specify the Jacobian matrix. From the equation 5.8 and the chain rule we have:

$$J_2 = \frac{\partial e_{i,k}}{\partial \Xi_{iw}} = \begin{bmatrix} \dfrac{\partial e_{i,k}(3)}{\partial p'} \dfrac{\partial p'}{\partial \Xi_{iw}} \\ \dfrac{\partial e_{i,k}(4)}{\partial q'} \dfrac{\partial q'}{\partial \Xi_{iw}} \end{bmatrix}. \tag{5.9}$$

Now by calculating separately the partial derivatives we obtain:

$$\frac{\partial p'}{\partial \Xi_{iw}} = \begin{bmatrix} \frac{f_x}{m_z} & 0 & -f_x\frac{m_x}{m_z^2} & -f_x\frac{m_x m_y}{m_z^2} & f_x(1+\frac{m_x^2}{m_z^2}) & -f_x\frac{m_y}{m_z} \\ 0 & \frac{f_y}{m_z} & -f_y\frac{m_y}{m_z^2} & -f_y(1+\frac{m_y^2}{m_z^2}) & f_y\frac{m_x m_y}{m_z^2} & f_y\frac{m_x}{m_z} \end{bmatrix}, \tag{5.10}$$

$$\frac{\partial q'}{\partial \Xi_{iw}} = \begin{bmatrix} \frac{f_x}{g_z} & 0 & -f_x\frac{g_x}{g_z^2} & -f_x\frac{g_x g_y}{g_z^2} & f_x(1+\frac{g_x^2}{g_z^2}) & -f_x\frac{g_y}{g_z} \\ 0 & \frac{f_y}{g_z} & -f_y\frac{g_y}{g_z^2} & -f_y(1+\frac{g_y^2}{g_z^2}) & f_y\frac{g_x g_y}{g_z^2} & f_y\frac{g_x}{g_z} \end{bmatrix}, \tag{5.11}$$

$$\begin{cases} \dfrac{\partial e_{i,k}(3)}{\partial p'} = \begin{bmatrix} f_{p'x}, f_{p'y} \end{bmatrix}, \\ \dfrac{\partial e_{i,k}(4)}{\partial q'} = \begin{bmatrix} f_{q'x}, f_{q'y} \end{bmatrix}, \end{cases} \tag{5.12}$$

$$\begin{cases} f_{p'x} = \dfrac{(x_p - x_q)||\vec{qp}||\,||\vec{qp'}|| - ((x_p-x_q)(x_{p'}-x_q)+(y_p-y_q)(y_{p'}-y_q))||\vec{qp}||(x_{p'}-x_q)/||\vec{qp'}||}{||\vec{qp}||^2||\vec{qp'}||^2}, \\ f_{p'y} = \dfrac{(y_p - y_q)||\vec{qp}||\,||\vec{qp'}|| - ((x_p-x_q)(x_{p'}-x_q)+(y_p-y_q)(y_{p'}-y_q))||\vec{qp}||(y_{p'}-y_q)/||\vec{qp'}||}{||\vec{qp}||^2||\vec{qp'}||^2}, \\ f_{q'x} = \dfrac{(x_p - x_q)||\vec{qp}||\,||\vec{pq'}|| - ((x_p-x_q)(x_{q'}-x_p)+(y_p-y_q)(y_{q'}-y_p))||\vec{qp}||(x_{q'}-x_p)/||\vec{pq'}||}{||\vec{qp}||^2||\vec{qp'}||^2}, \\ f_{q'y} = \dfrac{(y_p - y_q)||\vec{qp}||\,||\vec{pq'}|| - ((x_p-x_q)(x_{q'}-x_p)+(y_p-y_q)(y_{q'}-y_p))||\vec{qp}||(y_{q'}-y_p)/||\vec{pq'}||}{||\vec{qp}||^2||\vec{qp'}||^2}. \end{cases} \tag{5.13}$$

Thus, by combining the equations 5.9-5.13 the Jacobian matrix can easily be derived.

Similarly to the previous mentioned error function, a set of line segments is considered as an outlier only if $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 7 \cdot 10^{-5}$, where $\Sigma_{e_{i,k}}$ is again the identity matrix.

This threshold was again set empirically performing well in all our experiments. It can be noticed that its value is significantly lower compared to the associated threshold concerning the euclidean distance, a fact which is quite reasonable since it expresses angle distance in radians. Due to the increased uncertainty of this error function, the value of the threshold that produced the best performance for the algorithm is quite restrictive as even if for example $e_{i,k}(3)$ and $e_{i,k}(4)$ are equal to 0.4 degrees each, the matching pair will be considered an outlier.

5.1.3 Euclidean+Angle distance

Finally, we fuse the two previous methods and the emerging error function includes all the re-projection errors illustrated in Figure 5.1. Thus the new error function is defined as:

$$e_{i,k} = \begin{bmatrix} l_{i,k} \cdot {}^{h}\pi(\Xi_{iw}, P_{w,k}) \\ l_{i,k} \cdot {}^{h}\pi(\Xi_{iw}, Q_{w,k}) \\ \dfrac{q_{i,k}\vec{p}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|q_{i,k}\vec{p}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} - 1 \\ \dfrac{p_{i,k}\vec{q}'_{i,k} \cdot q_{i,k}\vec{p}_{i,k}}{\|p_{i,k}\vec{q}'_{i,k}\|\|q_{i,k}\vec{p}_{i,k}\|} + 1 \end{bmatrix}. \tag{5.14}$$

Having already defined individually the Jacobians for each error function in the equations 5.7 and 5.9, the associated Jacobian can effortlessly emerge as the combination of both. Particularly,

$$J = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}. \tag{5.15}$$

Regarding this error function, we adopt a slightly different strategy for discarding outliers. The main problem is that the two fused functions express different distances and thus their values are not in the same order of magnitude and are not normalised. In order to deal with this problem, by using the inverse of the covariance matrix which practically plays the role of a weighting matrix, we normalize the value of each component of the error function according to the previously established thresholds, so as to reach the value of 1 when becoming an outlier as set in the previous two cases. Specifically, the inverse covariance matrix is formulated as:

$$\Sigma^{-1}_{e_{i,k}} = \begin{bmatrix} \dfrac{1}{7.8} & 0 & 0 & 0 \\ 0 & \dfrac{1}{7.8} & 0 & 0 \\ 0 & 0 & \dfrac{1}{7 \cdot 10^{-5}} & 0 \\ 0 & 0 & 0 & \dfrac{1}{7 \cdot 10^{-5}} \end{bmatrix}. \tag{5.16}$$

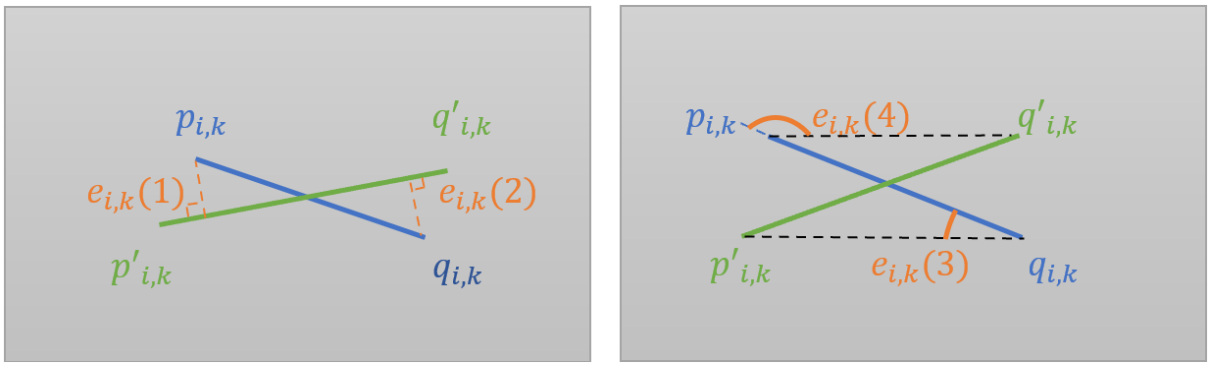


Figure 5.1: Re-projection errors of line segments. The left image depicts the Euclidean distance and the right one the angle distance.

This way, not only can we discard outliers much more efficiently by using information from both the pixel and the angle distance, but also the two individual error functions are weighted accordingly so that they both almost equally contribute to the bundle adjustment process. The threshold for detecting outliers is experimentally set to 1.5 (i.e. $e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k} > 1.5$).

As will be thoroughly explained in chapter 6, according to our experiments the most accurate error function is the one that uses the Euclidean distance exclusively, while in addition being much more efficient in terms of computational time as compared to the one that uses both Euclidean and angle distances. For these reasons, we adopted this function in the algorithm that uses both point and line features.

5.2 Motion-Only and Local Bundle Adjustment

The system performs motion-only bundle adjustments in the tracking thread so as to optimize only the pose of the camera, and a local bundle adjustment in the local mapping thread, where it optimizes both the pose of a local window of keyframes and the position of the features in the world. In both circumstances, the method of Levenberg-Marquardt (2.2.5) implemented in g2o [72] is used for the optimizations. Similarly to ORB-SLAM3 [1], the algorithm runs 4 times with 10 iterations each time for the motion-only bundle adjustment, and 2 times for the local bundle adjustment, with 5 iterations the first time, and 10 iterations the second time excluding the outliers detected in the first.

For the fusion of point and line features in the bundle adjustment, we introduce a novel adapting factor that weighs the involvement of the lines in the process. This requirement emerges from the fact that the error function associated with the line features is more biased in comparison to the one used for the point features. In particular, by minimizing the Euclidean distances between the endpoints of the re-projected 3D line segment and its corresponding infinite line in the image plane, we erroneously assume

that the points corresponding to the endpoints of the re-projected line are the ones that stand in its corresponding infinite line and simultaneously in the closest distance. In fact, the actual correct corresponding points cannot be totally retrieved as none line segment detector can reproduce exactly the same line segments with stable endpoints when observing a scene from different points of view. Concurrently, the accuracy of ORB-SLAM3 (regarding the visual stereo mode) is significantly biased in scenes in which few point matches are detected, in contrast to its remarkable efficacy in environments where the system determines an adequate number of point correspondences. So, the philosophy of the adapting factor is that as the number of point inliers elevates, the involvement of the line features reduces, and when the number of point inliers is small, then the lines are weighted equally to the points.

5.2.1 Motion-Only Bundle Adjustment

Assuming all the definitions made in the previous subsection and if $e_{i,j}$ stands for the projection error of the j-th map point in the i-th frame defined exactly the same as in ORB-SLAM2 [31], and ω is the vector that contains the variable to be optimized which is the pose of the i-th frame (i.e. $\{R_{iw}, t_{iw}\}$), then for the point-line algorithm the pose of the camera can be estimated as:

$$\omega^* = \underset{\omega}{\operatorname{argmin}}\{\sum_{j\in\mathbb{P}} \rho(e_{i,j}^T \Sigma_{e_{i,j}}^{-1} e_{i,j}) + \sum_{k\in\mathbb{L}} \rho(F(\alpha(\mathbb{P}))e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k})\}, \tag{5.17}$$

where $\mathbb{P}$ and $\mathbb{L}$ are the groups of the sets of point and line matches, respectively, $\rho(.)$ is the robust Huber cost function, $\alpha(\mathbb{P})$ is the total number of elements of group $\mathbb{P}$, div is an operator which gives the integer part of a division, and $F(\alpha(\mathbb{P}))$ is the adapting factor which is formulated as:

$$\mathrm{F}(\alpha(\mathrm{P})) = 2^{-\alpha(\mathbb{P})\mathrm{div}50} \tag{5.18}$$

The tuning of this adapting factor function has been established experimentally as explained in the next subsection. In the case of the only-line algorithm, the pose of the camera is similarly estimated but without including the projection errors of the points:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{k\in\mathbb{L}} \rho(e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k}). \tag{5.19}$$

5.2.2 Local Bundle Adjustment

In the local bundle adjustment, vector ω contains the pose of a group of covisible keyframes as well as the position of all the points and lines seen in those keyframes. If $\mathbb{K}_\mathbb{L}$ includes the previously mentioned group of keyframes and also the ones with which they observe common points and lines, defined as $\mathbb{P}_\mathbb{L}$ and $\mathbb{L}_\mathbb{L}$, respectively, then the optimization problem for the point-line algorithm is formulated as follows:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i \in \mathbb{K}_\mathbb{L}} [\sum_{j \in \mathbb{P}_\mathbb{L}} \rho(E_{i,j}) + \sum_{k \in \mathbb{L}_\mathbb{L}} \rho(E_{i,k})], \tag{5.20}$$

$$E_{i,j} = e_{i,j}^T \Sigma_{e_{i,j}}^{-1} e_{i,j}, \tag{5.21}$$

$$E_{i,k} = F(\alpha(\mathbb{P}_\mathbb{L})) e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k}. \tag{5.22}$$

Again, in the only-line algorithm, the local bundle adjustment does not contain the point features and is similarly defined as:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i \in \mathbb{K}_\mathbb{L}} [\sum_{k \in \mathbb{L}_\mathbb{L}} \rho(e_{i,k}^T \Sigma_{e_{i,k}}^{-1} e_{i,k})]. \tag{5.23}$$

5.3 Tuning of the Adapting Factor

For the tuning of the adapting factor, we assess the performance of the system for various values of the threshold (T) and weight (W) parameters in the general form of the adapting factor's equation: $F(\alpha(\mathbb{P})) = W^{-\alpha(\mathbb{P}) div T}$. In particular, we used six different values for each parameter and created a 6x6 grid structure to compare the results. For the evaluation, we chose two sequences from the EuRoC dataset[73]. Our first selection was MH04 which is a sequence that mostly consists of well textured and few dark low textured scenes, and our second one was V203 which is a particularly challenging sequence because it includes severe motion blur and therefore in many frames few point matches are detected from the system. Our main goal was to tune the factor on both well textured scenarios, such as MH04, in which the addition of lines increases the bias of the system, as well as on challenging scenarios in which line features can be vital and significantly improve system's efficacy. For every set of values, we run the sequences ten times each and the results emerge as the average of the twenty executions. The results are presented in Table 5.1 and Figure 5.2.

Threshold \ Weighting factor	1.25	1.5	1.75	2	2.25	2.5
30	0.169	0.158	0.159	0.142	0.158	0.168
40	0.184	0.145	0.139	0.162	0.158	0.183
50	0.213	0.164	0.184	0.131	0.137	0.162
60	0.188	0.172	0.159	0.150	0.175	0.167
70	0.204	0.166	0.150	0.166	0.137	0.147
80	0.257	0.159	0.170	0.176	0.138	0.165

Table 5.1: System's performance for various values of threshold and weight parameters of the adapting factor (RMSE ATE in m).

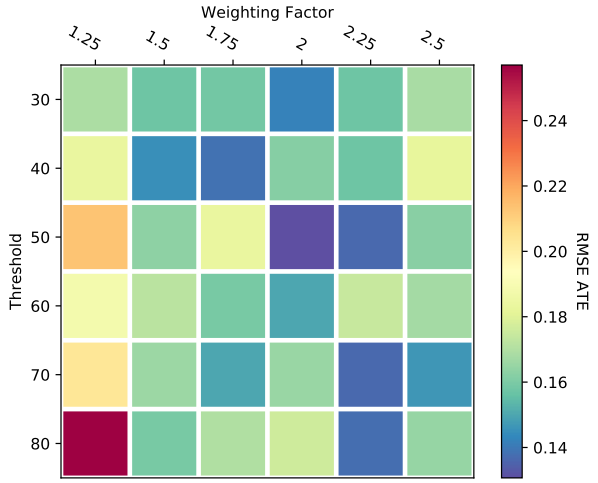


Figure 5.2: Heatmap visualising the results presented in Table 5.1.

The idea behind the tuning of the threshold parameter was to first start from the value of 30, which is the threshold of point matches for ORB-SLAM3 to consider that the tracking of the local map is successful, and then increase this value until no improvement of the results is possible for any value of weight parameter. The values of weight parameter were chosen experimentally, so that the system performs more accurately for the previous threshold values. From the results obtained, it can be concluded that the adapting factor performs comparably well for a broad range of values and is thus quite tolerant to changes. Specifically, the algorithm achieved the best performance for threshold equal to 50 and weight factor equal to 2, yet for many sets of values the system performs with similar accuracy. Between the threshold and weight parameters there is an expected connection. As threshold increases and weight decreases, the system verges on pure PL-SLAM which does not include the addition of the adapting factor, and when vice versa, the algorithm

approaches ORB-SLAM3. Besides, a pattern between the accuracy of the algorithm and the parameters can be observed. In particular, for a fixed threshold value, as weight increases, RMSE ATE abates, reaches a minimum, and then increases. This is clearly illustrated from the RMSE ATE acquired for a threshold value of 60. As expected from the previous comments, for lower threshold values the minimum can be observed mainly for a lower weight and vice versa. The previous observations are not absolutely confirmed from all the results due to the bias added from the non-deterministic nature of the system, however it is evident that a strong correlation between the previous comments and the results does exist.

Chapter 6

Evaluation

In this section we assess the performance of the only-line algorithm alone and the proposed point-line system in comparison to ORB-SLAM3[1] and PL-SLAM[3]. Concerning PL-SLAM, we present the published results from the authors on the EuRoC dataset[73] as the provided open-source code is incomplete and, hence, could not be executed. The metrics we used for the evaluation of the algorithms are the RMSE of the relative translational and the absolute trajectory error (ATE) presented in [12]. The relative translational error gives insight toward the local accuracy of the estimated trajectory as specifically relates to the drift of the trajectory over a fixed time interval. This metric is mostly recommended for evaluating visual odometry, in contrast to ATE which measures the global consistency of the estimated trajectory, and except of estimating the difference between the two trajectories, by visualizing the results, provides significant information about the accumulated error and the loop closures. Our decision to use both for the evaluation on EuRoC, yields to comparing our method with PL-SLAM, whose authors demonstrate results only regarding the relative translational error. For both metrics, we compare the estimated trajectory with all the poses included in ground truth. Concerning the relative error, as it is reasonable from the frequency of the camera used for EuRoC (20Hz), the value of the fixed time interval is 0.05s, and just for some pairs is 0.1s. This discrepancy is due to the fact that the evaluation algorithm can not determine a correspondence for some poses in the estimated trajectory, as their associated timestamps are not close enough to a timestamp of a pose in the ground truth trajectory according to the threshold of 0.02s used in [12]. Regarding the ATE, all the estimated trajectories are aligned with ground truth using a SE(3) transformation. All experiments have been executed on a laptop computer with an Intel Core i9-9980 CPU and 32 GB RAM. To account for the non-deterministic nature of the multi-threading systems, all the results presented in the tables of this section represent the average of ten executions for each sequence. Our open-source implementation includes all the necessary material to run the system in all the sequences presented in this section.

6.1 Evaluation of ONLY-LINE-SLAM

For the evaluation of the method that uses line features exclusively, we assess its performance on the EuRoC dataset[73] comparatively for all three different error functions discussed in Chapter 5. The results are depicted in Table 6.1 and Figure 6.1.

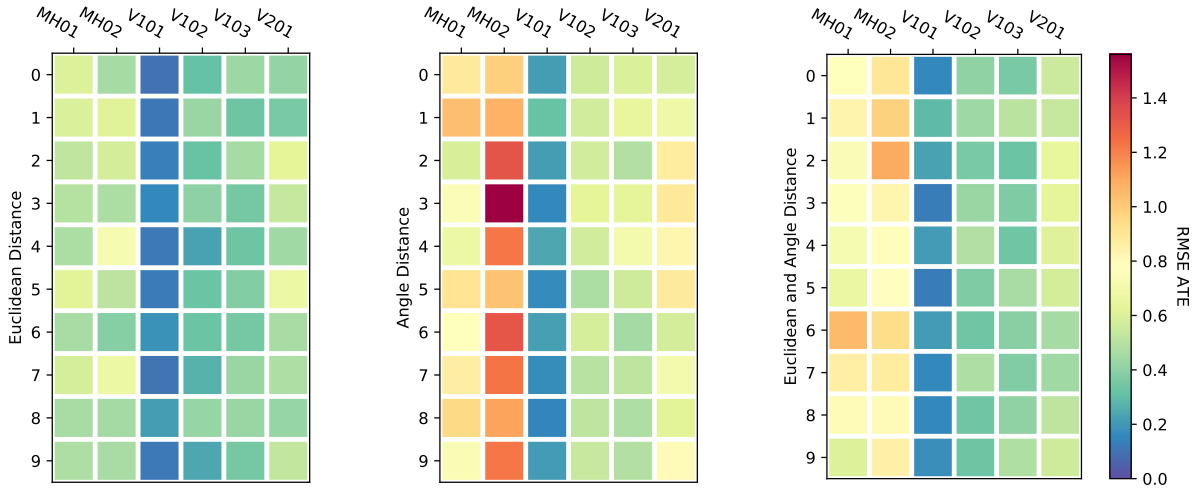


Figure 6.1: Heatmaps visualising the results obtained with the ONLY-LINE method executed on sequences of the EuRoC dataset. The left heatmap corresponds to the Euclidean distance error function, the middle one to the Angle distance and the right one to the combination of both. Each colored square depicts a single execution.

Table 6.1: Performance comparison of different error functions used in the ONLY-LINE method, executed on sequences of the EuRoC dataset (RMSE ATE in m).

Sequence	Euclidean	Angle	Euclidean and Angle
MH01 easy	**0.529**	0.817	0.782
MH02 easy	**0.534**	1.207	0.881
V101 easy	**0.138**	0.208	0.187
V102 medium	**0.328**	0.552	0.398
V103 difficult	**0.386**	0.558	0.403
V201 easy	**0.496**	0.744	0.554
Total Average	**0.402**	0.681	0.534

We present results for six specific sequences of the EuRoC dataset, as on the rest of the sequences the system achieved poor performance (RMSE ATE greater than 1.5m) for all three error functions, or even failed to complete (in the case of V203) due to severe motion blur. It is evident that the method which uses exclusively the Euclidean distance in the error function accomplishes the best performance. Concerning the angle distance error function, although it achieves to complete the presented sequences, it is associated to higher pose drift. Finally, the fusion of the aforementioned error functions increases the efficiency as compared to the one that uses the angle distance, but it is more biased than the one that uses exclusively the Euclidean distance. This performance was expected as

by adding a less efficient error function some bias is added to the pose estimation and concurrently, due to the extension of the error function, the required computing time increases. For these reasons, we selected to use the distance error function for the fusion of point and line features presented in the next section. The only advantage that the fusion method provides is that the addition of the angle error gives us the ability to reject outliers more efficiently during the optimization process, as both the information about distance and angle error is used.

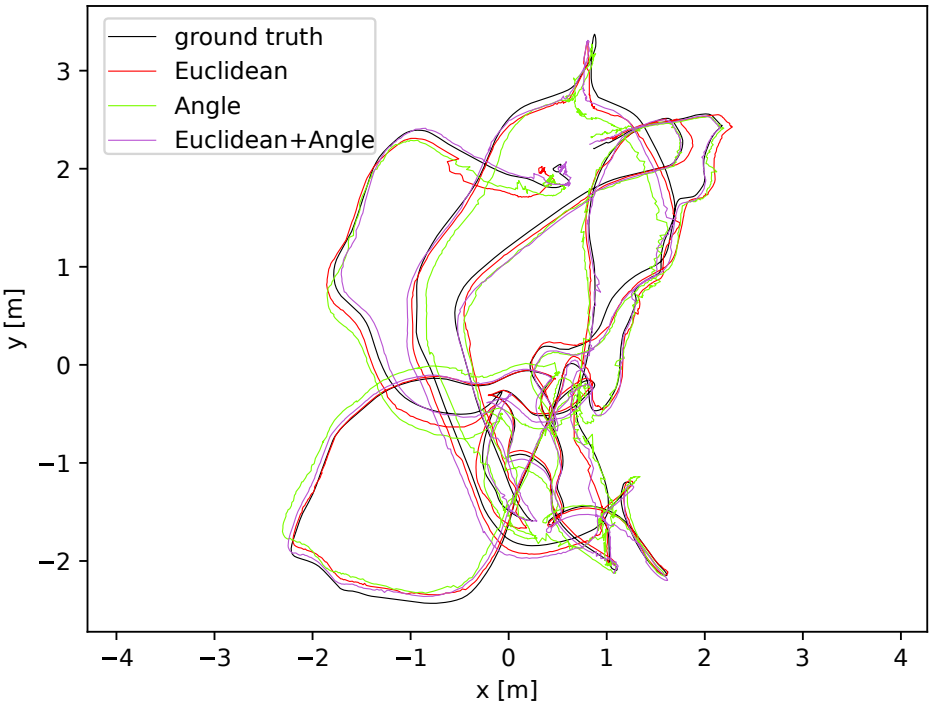


Figure 6.2: Estimated trajectories for the three different error functions on V101. The plot illustrates the fifth best execution of the algorithms.

The highest errors for all three methods are observed on the sequences with the longer duration. This happens mostly because, as we explained in section VI, line features are associated with relatively high pose drift, meaning that for longer sequences this drift is constantly being integrated leading to poorer results. For this reason, on medium or difficult sequences with shorter duration, like V102 or V103, the performance is superior compared to the one achieved on easier but longer sequences such as MH01 and MH02. The most impressive fact about the presented results is that the algorithm does not contain a loop closing process. Even the most precise state-of-the-art meth-

ods such as ORB-SLAM3 close loops, sometimes repeatedly within a sequence, so as to achieve trajectory errors in the region of centimeters to decimeters. Such sequences in which ORB-SLAM3 closes loops consistently are V102, V103 and V201. If the proposed ONLY-LINE-SLAM algorithm used an effective loop closing thread, we believe that its performance would surpass many of the-state-of-the-art point SLAM methods and that it would most probably approach the global performance of ORB-SLAM3.

6.2 Evaluation of ORB-LINE-SLAM on EuRoC Dataset

For the assessment of ORB-LINE-SLAM we firstly evaluate its performance on all eleven sequences of the EuRoC dataset. Table 6.2 shows the results obtained regarding the performance of our system compared with ORB-SLAM3 and PL-SLAM. Additionally, this table also presents the average coverage that ORB-LINE-SLAM and ORB-SLAM3 achieved on the V203 sequence. Coverage is defined as the fraction of localized frames with respect to the total number of ground truth frames in the sequence. For the rest of the sequences, both algorithms achieved 100% coverage. Except for the quantitative evaluation, it must be mentioned that our method also builds a physically particularly more meaningful representation of the environment, just as depicted in Figure 6.3. At the end of this section, we present numerous plots which visualize the results of Table 6.2.

Table 6.2: Performance comparison on EuRoC dataset. t_{rel} expresses RMSE relative translational error in m, and t_{abs} RMSE ATE in m.

Sequence	ORB-LINE-SLAM		ORB-SLAM3		PL-SLAM
	t_{rel}	t_{abs}	t_{rel}	t_{abs}	t_{rel}
MH01 easy	**0.038**	**0.033**	0.039	0.035	0.042
MH02 easy	**0.043**	0.019	**0.043**	0.019	0.052
MH03 medium	0.074	0.027	0.075	**0.025**	**0.040**
MH04 difficult	0.068	**0.094**	0.068	0.111	**0.064**
MH05 difficult	**0.064**	**0.049**	**0.064**	0.054	0.070
V101 easy	**0.023**	**0.033**	**0.023**	0.035	0.042
V102 medium	0.054	0.022	0.056	**0.021**	**0.046**
V103 difficult	**0.046**	**0.044**	0.048	0.059	0.069
V201 easy	**0.025**	**0.029**	**0.025**	0.036	0.061
V202 medium	**0.051**	**0.024**	0.052	0.044	0.056
V203 difficult	**0.066**	**0.167**	0.076	0.453	0.126
Coverage	**92%**		88%		
Total Average	**0.050**	**0.049**	0.052	0.081	0.061

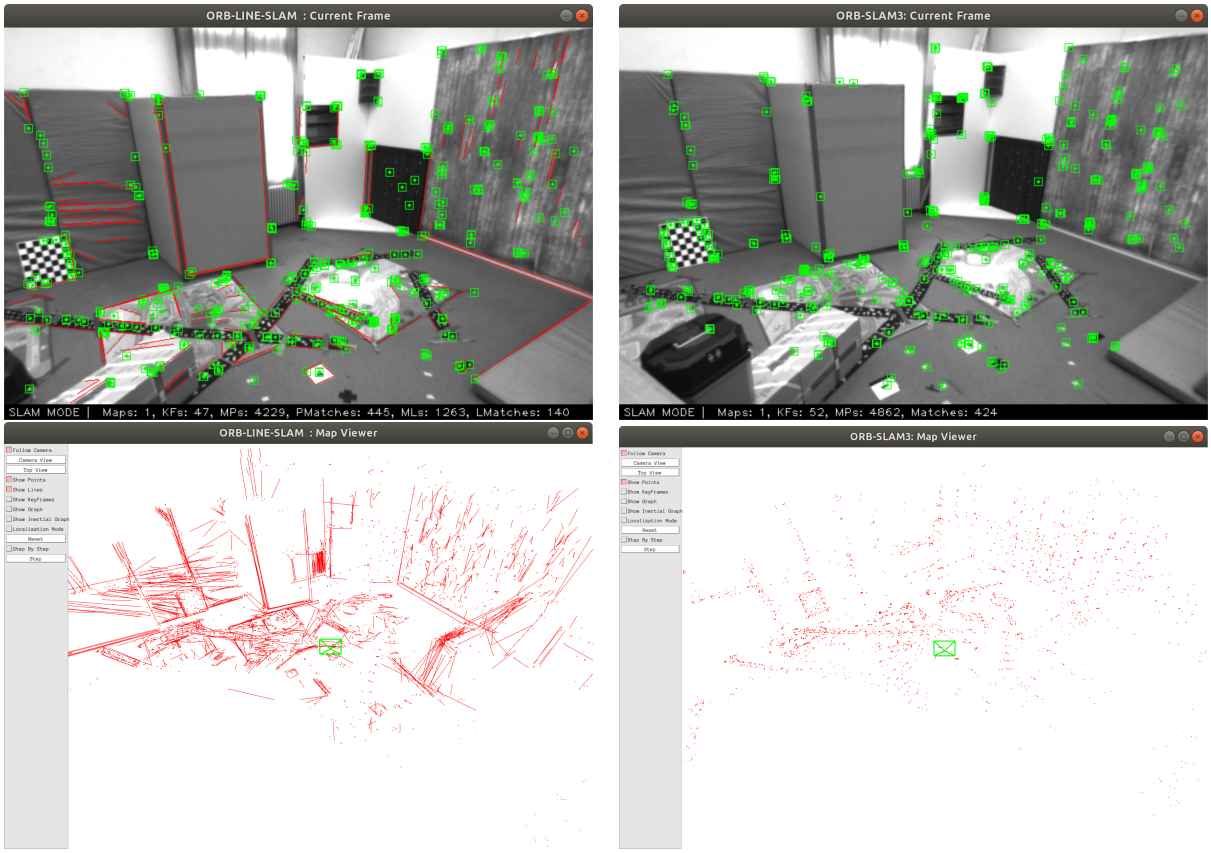


Figure 6.3: Map reconstruction by ORB-LINE-SLAM (left) and ORB-SLAM3 (right) of the same scene.

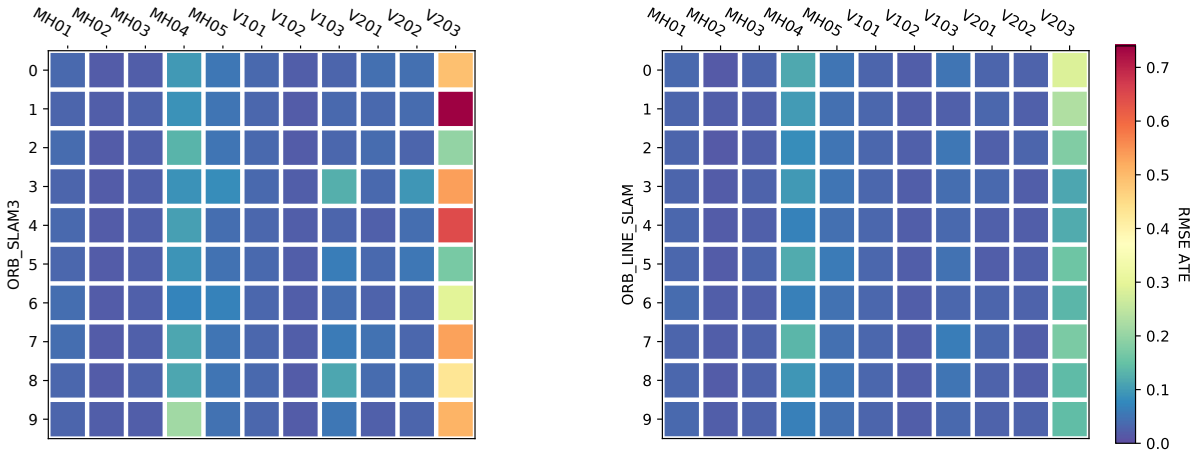


Figure 6.4: Heatmaps that represent all the executions for the EuRoC dataset. The left heatmap corresponds to ORB-SLAM3 and the right one to ORB-LINE-SLAM. Each colored square depicts a single execution.

As shown in Table 6.2 and Figure 6.4, it is evident that the addition of line features to ORB-SLAM3 significantly increases its robustness on the EuRoC dataset, especially on V203 which is a particularly challenging sequence that contains severe motion blur with some frames missing from one of the cameras. More specifically, concerning the aforementioned sequence, the ORB-LINE-SLAM algorithm proposed in this paper accomplishes consistently much lower trajectory and relative errors, and simultaneously higher coverage. The coverage that the two systems achieve varies from 80% to 94% for ORB-SLAM3 and from 89% to 97% for the proposed algorithm. The reason behind these results is that due to severe motion blur, in some frames of V203, ORB-SLAM3 determines small numbers of point matches and consequently the tracking gets lost, or the pose estimate is significantly biased as it is considerably affected from possible mismatches that gain high importance because of the limited number of total matches. However, due to the insertion of line features, the proposed system manages to determine accurately the pose of the camera on frames where ORB-SLAM3 always gets lost, and to consequently attain higher precision. Moreover, at the last turn of the sequence, ORB-LINE-SLAM manages to recover the pose of the camera earlier than ORB-SLAM3 and therefore achieves to close a loop much more frequently. All the previous comments are illustrated in Figures 6.5 and 6.6.

Concerning V201 and V202, our system consistently achieves better accuracy by approximately one to two centimeters RMSE ATE (Figure 6.8), while on V103 provides more stable results compared to ORB-SLAM3. Particularly, in some executions of V103, the absolute errors obtained by running ORB-SLAM3 were higher than ten centimeters in contrast to our system which seems to produce low errors more steadily, a fact that justifies the results presented in Table 6.2. Additionally, regarding V101, our experiments indicate that our system is more precise by one to three millimeters. In most of the rest of the sequences, although the proposed method shows a tendency to be a little more stable and accurate, this cannot be stated with certainty as due to the non-deterministic nature of the multi-threading system the results obtained by the executions of the same sequences vary, and a difference in performance cannot be reliably deduced. Only on V102 sequence does ORB-SLAM3 manage to achieve consistently lower absolute errors by approximately one millimeter.

The same interpretations are also corroborated by the relative translational error. Particularly, in all the sequences, ORB-LINE-SLAM attains better or at least the same relative errors as ORB-SLAM3. As it is evident from Table 6.2 the relative errors between the two methods are fairly close, except V203 where, for the same reasons as with the ATE, our algorithm significantly outperforms ORB-SLAM3. These results were quite expected if we take into account that the lines contribute to the algorithm mostly when few point matches are determined. Hence, considering that almost all the sequences of EuRoC contain mostly well-textured scenes where the algorithm detects enough point correspondences, it is logical that the participation of lines plays a major role in pose

prediction only in few frames. Although their contribution is not clearly visible in many sequences through the relative error as they have a significant role only in few out of some thousand pairs of frames that are compared, their importance is undeniable if someone thinks about the severe consequences that can have a momentary culmination of the drift of an autonomous system. To adduce an example, because of such a spike in the drift error, may an autonomous vehicle can not pass from an aperture, collide with an obstacle, or even reach its destination with an error higher than the desirable for the application. This concept is clearly illustrated in Figure 6.7 where noteworthy spikes are apparent in the relative translational error of ORB-SLAM3 regarding V103, in contradiction to ORB-LINE-SLAM whose relative error is particularly smoother. For this reason, as we mentioned before, ORB-LINE-SLAM seems more stable and accurate on V103, a fact that is more perceptible through the ATE because a spike in the drift affects only one measurement in the relative error, but many in the ATE as this drift will not be corrected until the algorithm closes a loop.

Comparing the results of our system with the ones of PL-SLAM, it is discernible that ORB-LINE-SLAM significantly prevails over PL-SLAM. Specifically, PL-SLAM is superior only in 3 out of the 11 sequences, and in two of them just for some millimeters, while in MH03 in which is the most dominant, for 3.4 centimeters in terms of relative translational error. On the other hand, our algorithm outperforms PL-SLAM in the rest 8 sequences, and particularly in half of them, in scale of centimeters (V101 for 1.9cm, V103 for 2.3cm, V201 for 3.6 cm). The most substantial difference between the two methods is located on V203, which we remind is the most challenging sequence of EuRoC, where our system is associated with relative error 6 centimeters lower. The average error of the 11 sequences concerning our algorithm is 1.1 centimeter lower than PL-SLAM's.

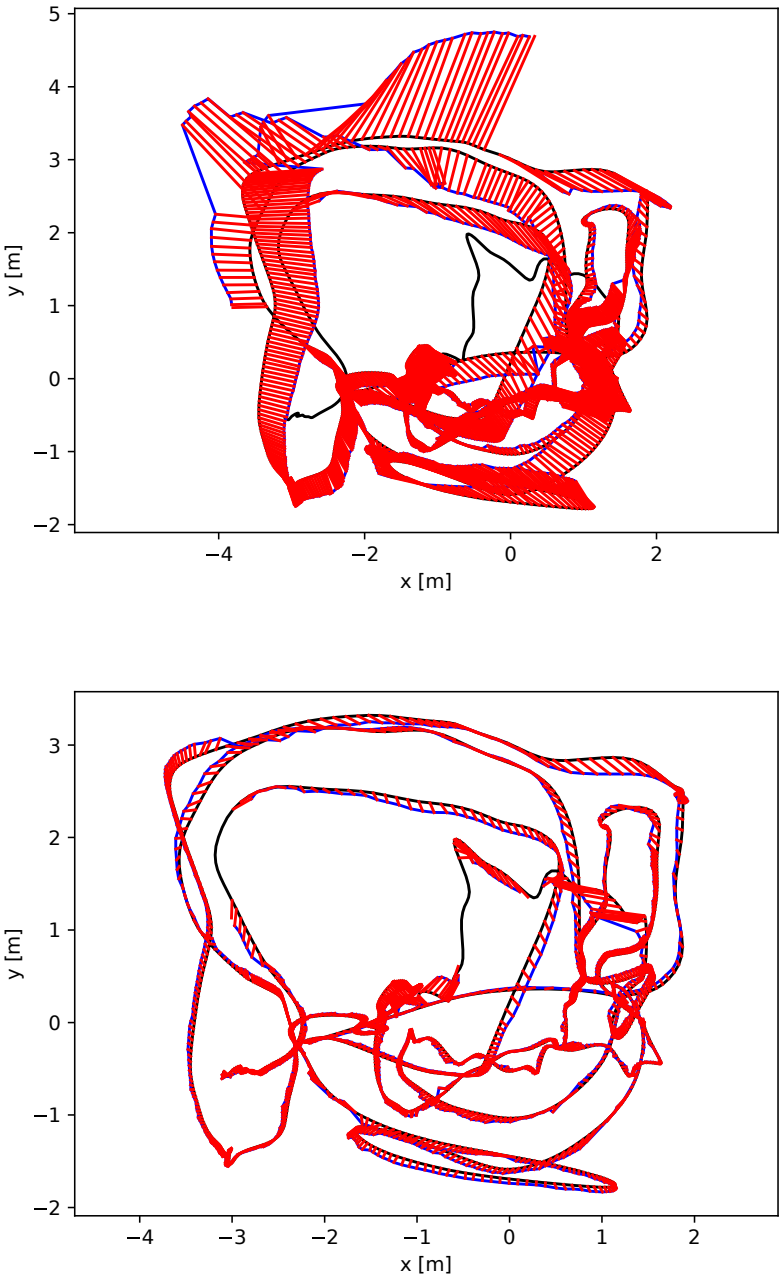


Figure 6.5: Estimated trajectory (blue), ground-truth trajectory (black) and difference between the pose of the corresponding frames (red) on V203. The upper plot is associated with the fifth best execution of ORB-SLAM3 and the one below with the fifth best of ORB-LINE-SLAM.

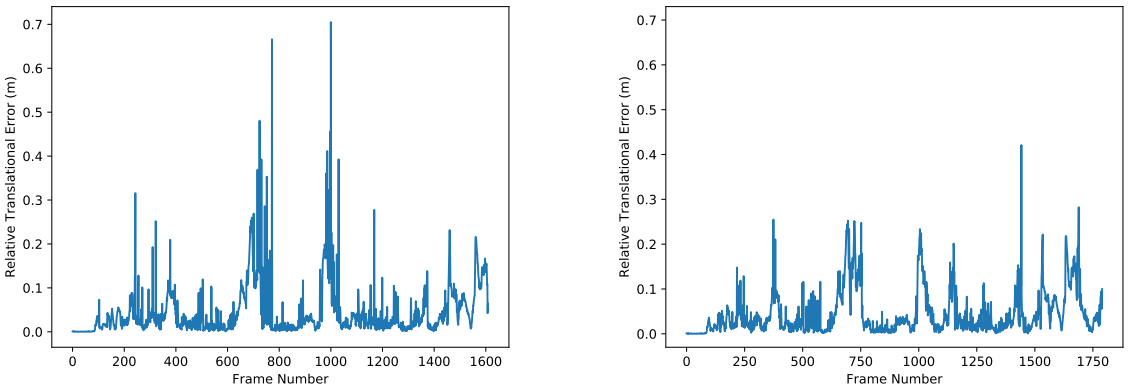


Figure 6.6: Relative translational error for ORB-SLAM3 (left) and ORB-LINE-SLAM (right) on V203. Both plots correspond to the fifth best execution.

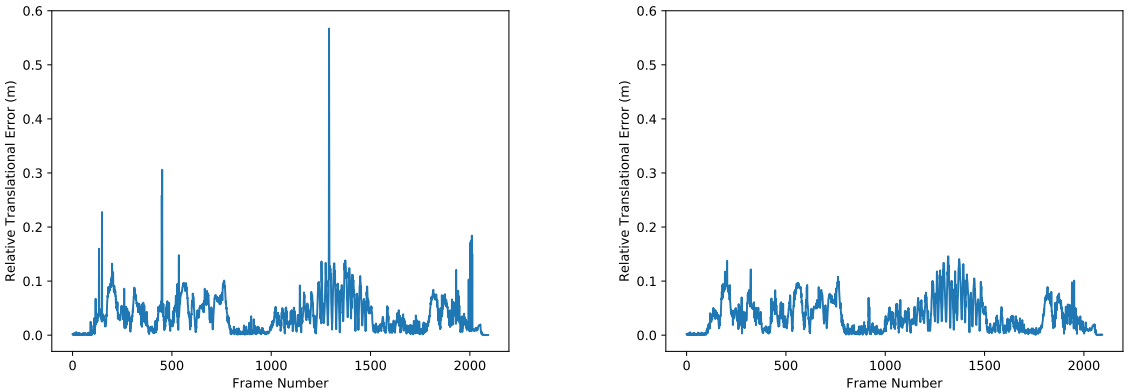


Figure 6.7: Relative translational error for ORB-SLAM3 (left) and ORB-LINE-SLAM (right) on V103. Both plots correspond to the fifth best execution.

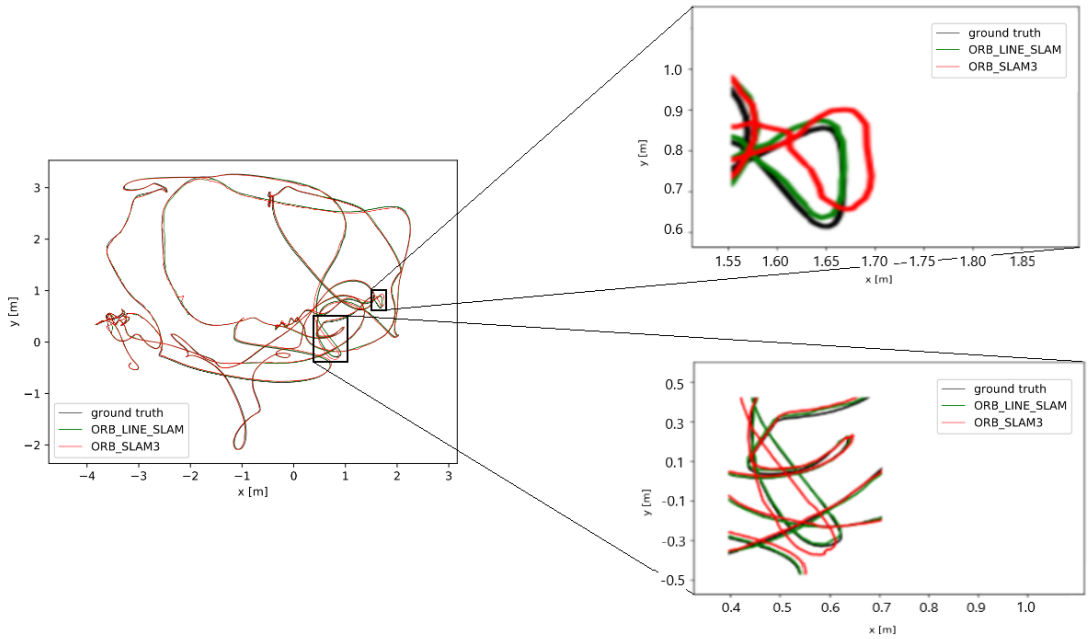


Figure 6.8: Estimated trajectory of ORB-SLAM3 and ORB-LINE-SLAM on V202. The plot illustrates the fifth best execution of the algorithms.

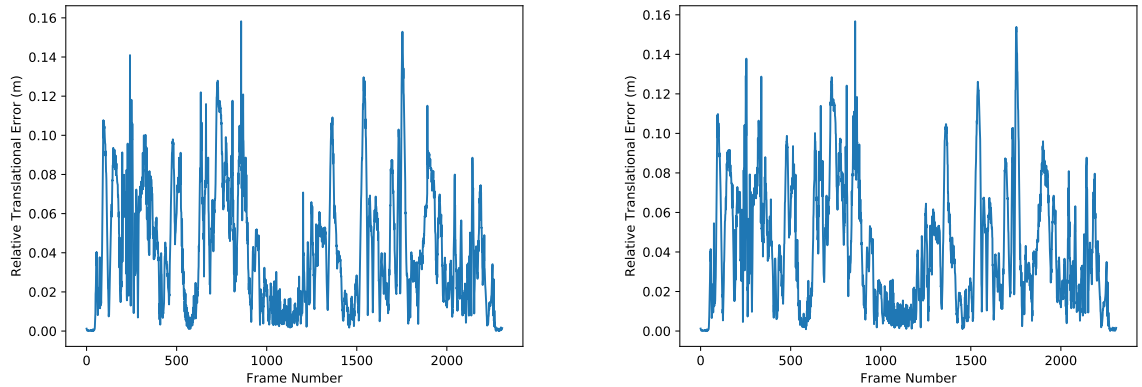


Figure 6.9: Relative translational error for ORB-SLAM3 (left) and ORB-LINE-SLAM (right) on V202. Both plots correspond to the fifth best execution.

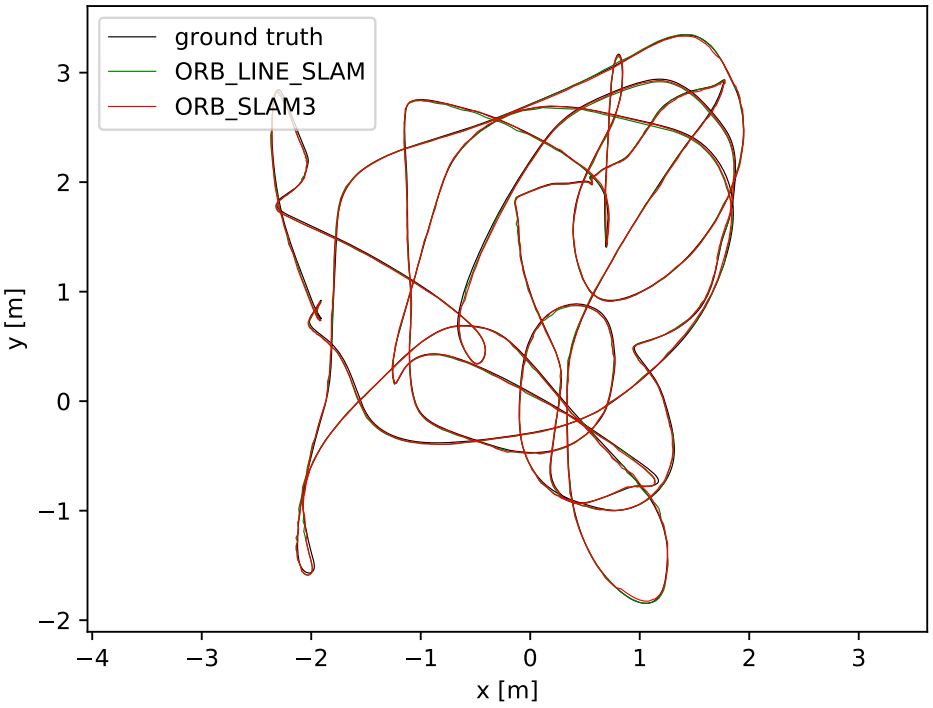


Figure 6.10: Estimated trajectory for the two algorithms on V102. The plot represents the fifth best executions.

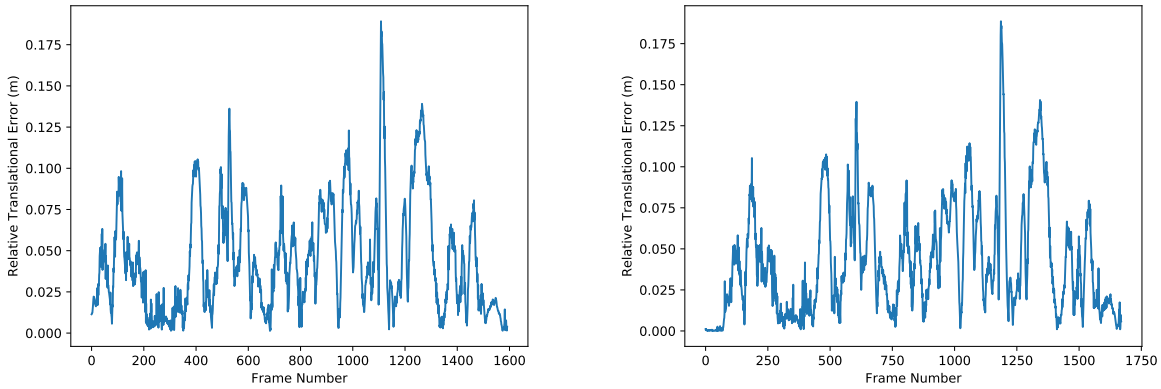


Figure 6.11: Relative translational error for ORB-SLAM3 (left) and ORB-LINE-SLAM (right) on V102. Both plots correspond to the fifth best execution.

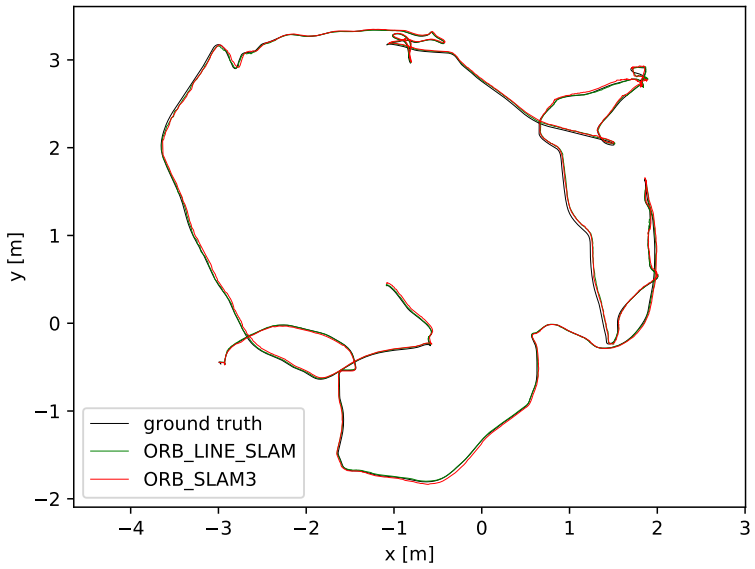


Figure 6.12: Estimated trajectory for the two algorithms on V201. The plot represents the fifth best executions.

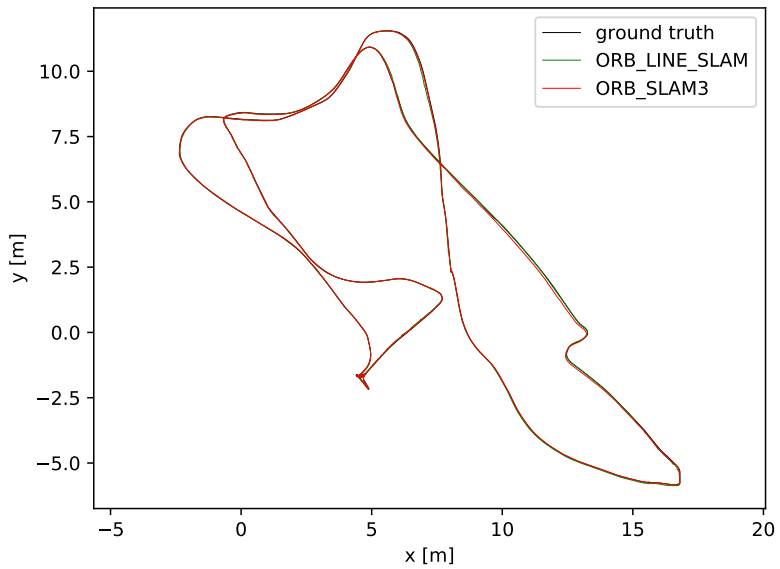


Figure 6.13: Estimated trajectory for the two algorithms on MH05. The plot represents the fifth best executions.

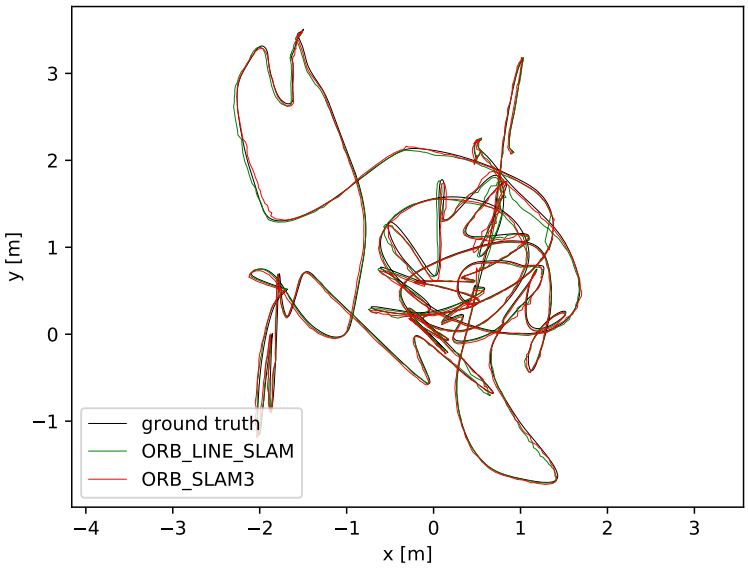


Figure 6.14: Estimated trajectory for the two algorithms on V103. The plot represents the fifth best executions.

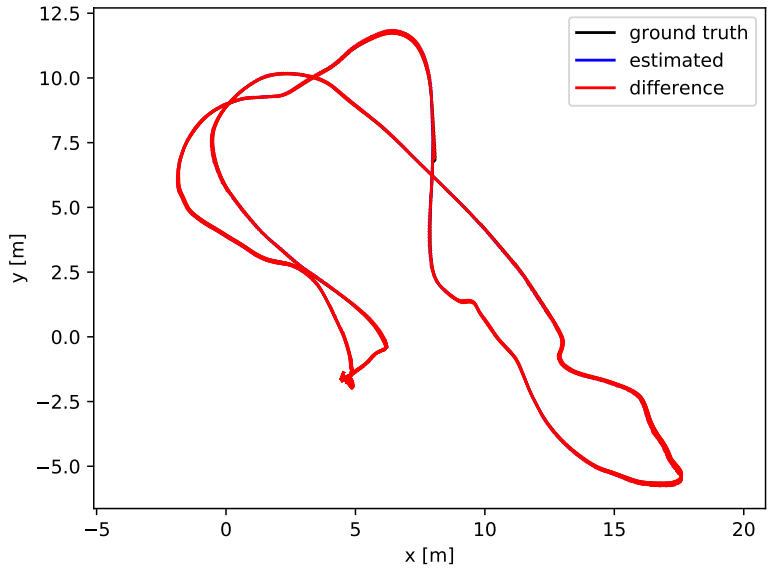


Figure 6.15: Estimated trajectory for ORB-LINE-SLAM on MH04. The plot represents the fifth best execution.

6.3 Evaluation of ORB-LINE-SLAM on UMA Dataset

To further assess the efficacy of our method, we evaluated its performance on two challenging sequences of the UMA dataset[74]. The UMA visual-inertial dataset contains many sequences that correspond to a variety of different scenarios, such as low-textured, indoor, and outdoor with constant illumination changes and sun exposure. Many of these sequences include some frames in which there is absolute absence of light or the sensor rotates extremely sharply, so that no matching features are detected and thus these sequences are more appropriate for the evaluation of visual-inertial systems. Also, two different stereo cameras are attached on the visual-inertial sensor. For our evaluation, we chose to use the custom stereo rig which was built from two IDS uEye UI-1226LE-M cameras, each providing 752x480 pixel monochrome images. Both sequences were selected to contain low textured and dark scenes so as to evaluate the efficiency of our method in comparison to ORB-SLAM3 in these challenging scenarios. Due to the fact that on all the UMA sequences the sensor explores an environment following a somewhat circular path and finally returns to the initial scene, and because the ground truth provided includes only some frames from the start and the ending of the sequence, in order to evaluate and compare the actual errors that the two methods achieve, we have deactivated the loop closing thread. If we were to assess the performance of the methods with the loop closing thread activated, both would close a loop at the ending of each sequence and the pose of the camera would be corrected according to the pose in the initial frames, and hence the error would be almost zero regardless of the drift of the sensor at the end of the sequence. Furthermore, assuming that the samples are independent with equal variances, we perform a t-test [75] on the results of the two algorithms to determine the significance of their difference in challenging conditions. Table 6.3 and Figure 6.16 compare the performance of the two systems on the two selected sequences (*hall1-rev-eng* and *corridor-eng*) of the UMA dataset.

Table 6.3: Performance comparison on UMA dataset (RMSE ATE in m). RMSE ATE represents the average of ten successful executions (± the standard deviation) and the percentages correspond only to the ten initial executions of each sequence.

Sequence	ORB-LINE-SLAM	ORB-SLAM3	P of t-test
hall1-rev-eng	**0.780 (±0.116)**	0.934 (±0.162)	0.025
Percentage of successful executions	**100%**	30%	
corridor-eng	**1.055 (±0.099)**	1.156 (±0.110)	0.044
Percentage of successful executions	**100%**	90%	

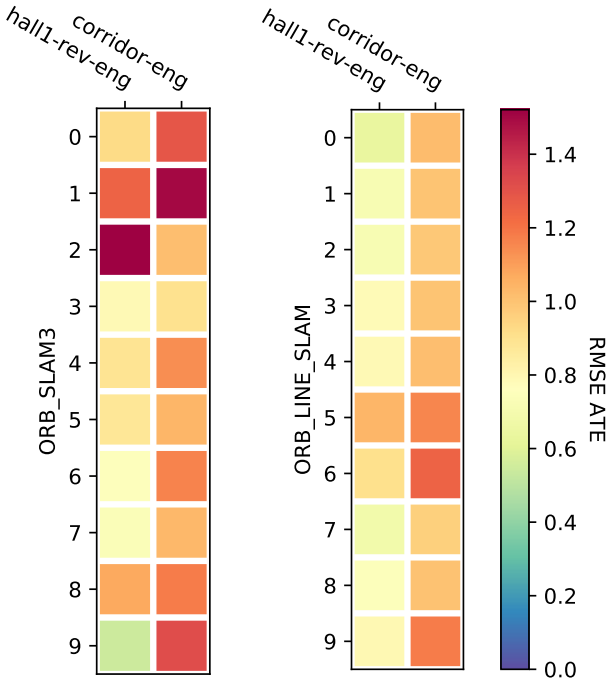


Figure 6.16: Heatmaps that represent all the executions for the UMA dataset. The left heatmap corresponds to ORB-SLAM3 and the right one to ORB-LINE-SLAM. Each colored square depicts a single execution.

As can be deduced from these results, the proposed ORB-LINE-SLAM system is both more precise and stable on low textured scenes and on sequences that include intense illumination changes. In particular, because the system detects matches from both line and point features, in dark and low textured scenes in which few point matches are determined, the proposed method manages to survive and provide reliable pose estimation, by exploiting information from the additional line matches, in contrast to ORB-SLAM3 which often gets lost and fails. This can also be observed in Figure 6.17, which shows comparative performance on two snapshots from the selected sequences. Except from these, as verified from the P values of the t-tests which are lower than 5%, our system shows consistency in obtaining lower errors. Particularly, these values indicate that there is a significant difference between the performance of the two methods and that the null hypothesis is false. Because the ground truth contains only some frames of the sequences, plots that compare the whole computed trajectory with the ground truth cannot be fully produced. However, trajectories and sparse reconstructions of the environment obtained on these two sequences using the proposed ORB-LINE-SLAM method are depicted in Figure 6.18.

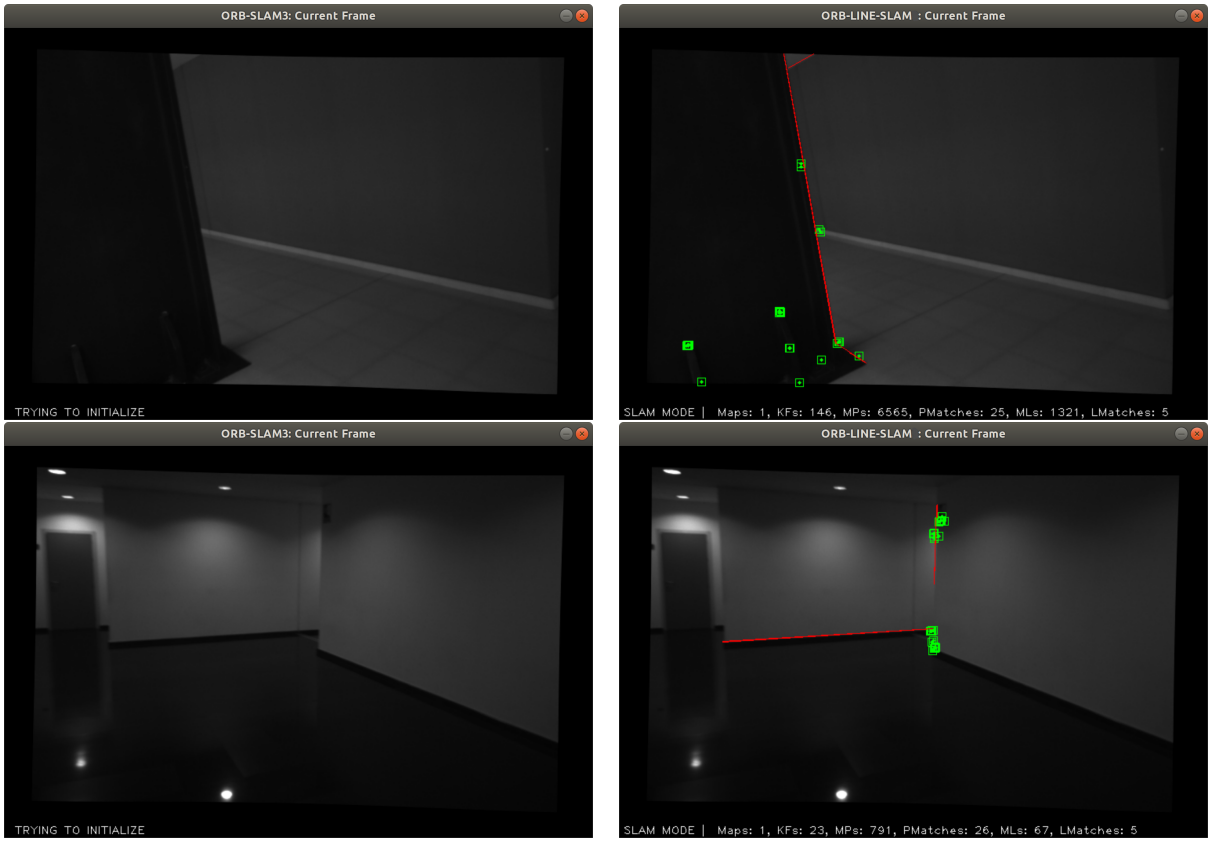


Figure 6.17: Snapshots from *hall1-rev-eng* and *corridor-eng*. The upper images correspond to *hall1-rev-eng* and the ones beneath to *corridor-eng*. The images on the left correspond to ORB-SLAM3 and the ones on the right to ORB-LINE-SLAM.

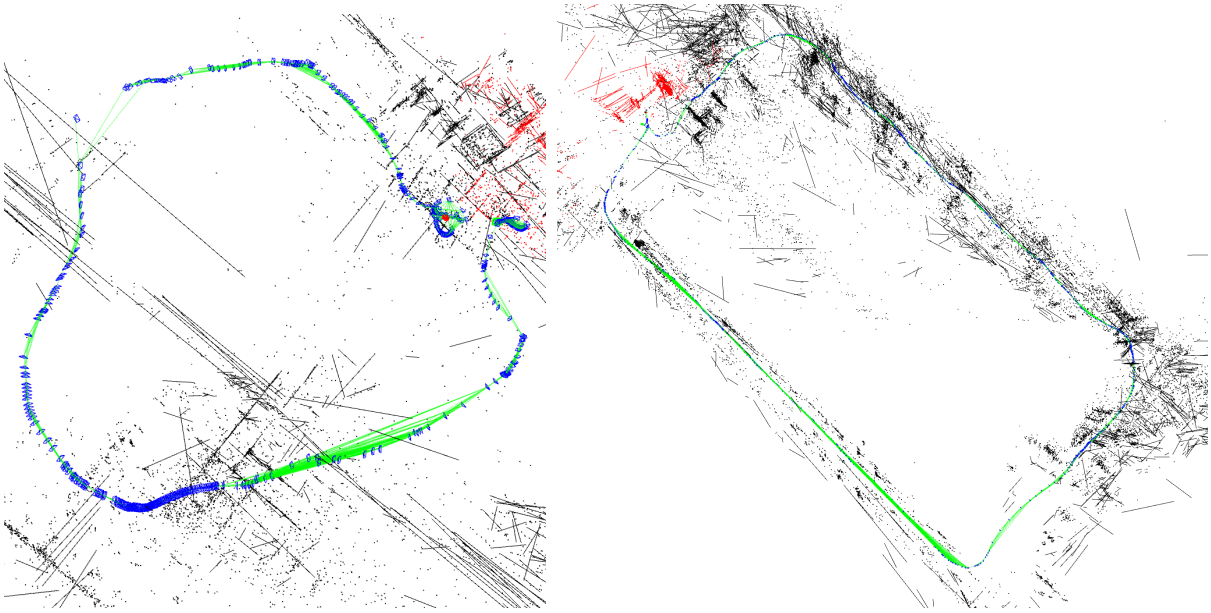


Figure 6.18: Trajectories and sparse reconstructions of the environment regarding *hall1-rev-eng* (left) and *corridor-eng* (right).

6.4 Computing Time

As a final step for the evaluation of the proposed system, we measured the running time of the main operations of the tracking and local mapping threads. Table 6.4 shows the results obtained for different sequences of EuRoC and UMA datasets. From these numbers, it can be inferred that our system is able to satisfy the timing requirements so as to operate in real-time, since the average total tracking time per frame is less than the inverse frames per second of the camera. However, as it was expected, the integration of line segments into ORB-SLAM3 elevated the computational burden, mostly due to the time required for the line detection process. Nevertheless, this hindrance can easily be overcome from our algorithm even in less powerful CPUs, as the user has the ability to tune the scale of the image from which the line segments are to be detected, thus reducing remarkably the computing time with only a small sacrifice on the number of detected lines.

Table 6.4: Running time of the tracking and local mapping threads of ORB-LINE-SLAM and ORB-SLAM3 (mean ± standard deviation in ms).

Settings	Sequence	V103	V203	corridor-eng
	Dataset	EuRoC	EuRoC	UMA
	Resolution	752 × 480	752 × 480	752 × 480
	Camera Frequency (Hz)	20	20	25

ORB-LINE-SLAM

Tracking	Feature Extraction	28.88 ± 6.62	29.27 ± 7.41	20.35 ± 5.76
	Stereo Matching	9.09 ± 2.87	9.30 ± 2.67	8.28 ± 2.58
	Pose Prediction	3.09 ± 1.20	3.08 ± 1.11	2.75 ± 1.00
	Local Map Tracking	7.43 ± 3.76	5.86 ± 3.58	6.06 ± 2.92
	New Keyframe Decision	0.05 ± 0.05	0.05 ± 0.09	0.04 ± 0.10
	Total	49.60 ± 11.01	48.93 ± 11.35	38.42 ± 11.02
Local Mapping	Keyframe Insertion	6.16 ± 1.76	5.45 ± 1.92	11.82 ± 6.02
	Map Point & Line Culling	0.35 ± 0.15	0.26 ± 0.10	0.14 ± 0.07
	Map Point Creation	32.75 ± 16.54	24.25 ± 12.86	35.26 ± 15.21
	Local Bundle Adjustment	95.20 ± 90.41	44.69 ± 46.78	85.96 ± 115.36
	Keyframe Culling	3.33 ± 2.66	1.93 ± 1.95	3.43 ± 2.80
	Total	144.29 ± 108.00	82.28 ± 59.25	148.67 ± 128.51

ORB-SLAM3

Tracking	Feature Extraction	11.68 ± 2.80	11.93 ± 3.94	10.43 ± 1.75
	Stereo Matching	8.41 ± 2.76	8.57 ± 2.93	7.39 ± 2.37
	Pose Prediction	2.01 ± 0.65	1.98 ± 0.81	1.75 ± 0.61
	Local Map Tracking	7.03 ± 3.10	5.13 ± 2.78	4.61 ± 2.21
	New Keyframe Decision	0.04 ± 0.08	0.04 ± 0.09	0.03 ± 0.08
	Total	30.20 ± 5.51	28.70 ± 6.97	25.03 ± 5.55
Local Mapping	Keyframe Insertion	5.91 ± 1.56	5.29 ± 2.00	10.05 ± 4.55
	Map Point & Line Culling	0.25 ± 0.10	0.23 ± 0.09	0.10 ± 0.04
	Map Point Creation	33.21 ± 15.48	21.67 ± 14.34	31.54 ± 13.08
	Local Bundle Adjustment	80.92 ± 77.15	40.34 ± 56.54	54.74 ± 63.70
	Keyframe Culling	4.34 ± 3.46	1.91 ± 2.78	2.38 ± 2.35
	Total	130.79 ± 95.62	74.95 ± 72.49	108.95 ± 75.88

Chapter 7

Conclusions

7.1 Brief Summary of Thesis

In this thesis we proposed ORB-LINE-SLAM, a hybrid visual SLAM system that has the ability to function both a) exclusively with lines or b) by adaptively combining points and lines together. We began by defining the problem and briefly the history of SLAM, and its possible impact on the modern world. Then, we outlined some of the challenges of contemporary SLAM and recommended feasible solutions through already existing methods and technology developed by the community. The introduction ends up with a discussion about the contribution of this thesis, as well as by summarizing its general structure.

The next chapter is devoted to presenting some of the most salient terminology, concepts, and mathematical tools regarding the problem of SLAM. Starting from simple terminology and continuing with the qualitative and mathematical description of core tools of SLAM, this part of the thesis aims to gently introduce the reader to the key ideas and components of SLAM and provide him with the pertinent foundation for the rest of the thesis.

Subsequently, we briefly overview the most prominent related work to the field. In chronological order, we first highlighted the original approaches which focused mainly on probabilistic methods, from *Bayesian filter* to *Extended Kalman filter* and beyond, and played a major role as the basis of contemporary SLAM. Continuing to the modern methods, we briefly explained the main aspects of the most famous *LiDAR SLAM* algorithm, and we concluded by giving emphasis to the feature-based approaches which mostly inspired our work.

Next, a detailed overview of our system is presented. The purpose of this chapter is to give in-depth knowledge to the reader about the operation of the system, as well as it also functions as a guide for the open-source code available in [64]. Our algorithm consists of 3 parallel threads, the *Tracking* thread, the *Local Mapping* thread, and the *Loop Closing* thread. The line segments are totally included only in the first two threads, while they just passively participate in the third. A detailed description of all the different components of these threads, as well as the various conditions and thresholds connected to the algorithm are defined in this chapter.

We continued with portraying the most crucial process for every SLAM system, the *Bundle Adjustment*. In this part of the thesis, we formulate the problem of *Bundle Adjustment* and give insight to the reader toward the different error functions and hidden in the code parameters used for the optimizations. Additionally, we introduce the most principal novelty of this thesis, which is a novel experimentally tuned adapting factor that aims to more efficiently fuse line and point features. The tuning of this parameter as well as the qualitative interpretation for the significance of its use are included in this chapter.

Finally, we evaluate our method by comparing it with the state-of-the-art ORB-SLAM3 [1] and PL-SLAM [3]. Moreover, we thoroughly compare different error functions

for the bundle adjustment, providing results and drawing conclusions about the comparative effectiveness of each one for the problem of visual SLAM. Our experiments indicate that our point-line algorithm remarkably outperforms PL-SLAM and improves the accuracy of ORB-SLAM3, especially in demanding sequences that contain low-textured scenes or significant motion blur. Many associated plots are also presented that justify these claims.

The contribution of this thesis can therefore be summarized as follows: first, a background tutorial for the key ideas and mathematical tools connected to SLAM. Second, the introduction of the first open-source line SLAM system which can solve the visual SLAM problem with reliable precision by using exclusively line features. Then, a novel strategy of weighted fusion of point and line features through the use of an experimentally tuned adapting factor. Additionally, a detailed comparison of the accuracy of the most renowned line segment detectors and error functions regarding the SLAM problem. Finally, the incorporation of the line segments into the stereo algorithm of ORB-SLAM3, and the creation of an open-source point-line system that outperforms the stereo version of ORB-SLAM3 which is currently the most accurate available stereo open-source implementation.

7.2 Future Extensions

According to our observations, we recommend that future expansion of this work should concentrate into two distinct directions. First, the most significant flaw of line SLAM systems, is the increased computational time along with the insufficient number of line correspondences and their unstable endpoints. To tackle these hindrances, more recent and effective open-source line descriptors based on machine learning, such as DLD[76] and WLD[77], could be used instead of LBD[69], so as to reinforce the process of line matching. Moreover, a line detector that would aim to the visual SLAM problem by focusing on finding repeatable lines with stable endpoints, instead of meaningfully reconstructing the surrounding environment, could definitely reduce the computational cost as well as increase the number of line correspondences. Such an efficient line detector that focuses on SLAM and also generates robust descriptors for matching is presented in [78].

The second direction that our algorithm could be extended relates to sensors that could be used instead of the stereo camera, or sensors that could be added to our system. Specifically, our work could be reinforced by including line features into the monocular, RGBD, and visual-inertial library of ORB-SLAM3. With the addition of an IMU, our algorithm would be significantly more robust in cases of significant motion blur, insufficient illumination, and low-texture scenes, in which the system can not define adequate number of matches to optimize the pose of the camera. Except for that, an innovative direction would be to combine a visual sensor with a LiDAR and create a hybrid visual-

LiDAR SLAM system that would take advantage of both sensors. For instance, a LiDAR can provide information about the structure of the scene even without illumination, and hence such a system would be much more tolerant in illumination changes, and in occasions where the light is absent. Finally, an efficient lightweight loop closing algorithm that would use both point and line features would be extremely beneficial for our system. Such an addition, would remarkably elevate the performance of our only-line method, as it would be able to close loops and thus correct the pose of the camera and the accumulated drift, which could render our method unsuitable for demanding applications that require high precision regarding the pose estimation.

Bibliography

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[2] J. Zhang and S. Singh, "LOAM : Lidar Odometry and Mapping in real-time," *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, 01 2014.

[3] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[4] I. El Bouazzaoui, S. Rodriguez, B. Vincke, and A. El Ouardi, "Indoor visual SLAM dataset with various acquisition modalities," *Data in Brief*, vol. 39, p. 107496, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340921007770

[5] "MS Windows NT Kernel Description," https://www.mathworks.com/help/vision/ug/camera-calibration.html, accessed: 2022-10-23.

[6] "MS Windows NT Kernel Description," https://slideplayer.com/slide/5162869/, accessed: 2022-10-23.

[7] F. Gu, Z. Song, and Z. Zhao, "Single-Shot Structured Light Sensor for 3D Dense and Dynamic Reconstruction," *Sensors*, vol. 20, p. 1094, 02 2020.

[8] C. Loop and Z. Zhang, "Computing rectifying homographies for stereo vision," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, 1999, pp. 125–131 Vol. 1.

[9] "MS Windows NT Kernel Description," https://www.cs.cmu.edu/~16385/s17/Slides/13.1_Stereo_Rectification.pdf, accessed: 2022-10-23.

[10] Du, M. Muslikhin, Hsieh, and Z. Wang, "Stereo Vision-Based Object Recognition and Manipulation by Regions with Convolutional Neural Network," *Electronics*, vol. 9, p. 210, 01 2020.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[13] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4503–4508.

[14] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features," *Sensors*, vol. 18, no. 4, p. 1159, 2018.

[15] S. Rady, "INFORMATION-THEORETIC ENVIRONMENT MODELING FOR MOBILE ROBOT LOCALIZATION," Ph.D. dissertation, 01 2012.

[16] C. Stachniss, G. Grisetti, and W. Burgard, "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters." in *Robotics: Science and systems*, vol. 2, 2005, pp. 65–72.

[17] J. Blanco, J. Fernández-Madrigal, and J. Gonzalez, "A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao—Blackwellized Particle Filters," *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 73–89, 2008.

[18] Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 1403–1410 vol.2.

[19] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[20] H. Kim, "SceneLib2 - MonoSLAM open-source library," https://github.com/hanmekim/SceneLib2.

[21] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[22] ——, "Improving the agility of keyframe-based slam," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 802–815.

[23] ——, "Parallel Tracking and Mapping on a camera phone," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 83–86.

[24] ——, "PTAM-GPL," https://github.com/Oxford-PTAM/PTAM-GPL, 2013.

[25] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[26] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942.

[27] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," https://github.com/tum-vision/lsdslam.

[28] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[29] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[30] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO," https://github.com/uzh-rpg/rpg_svo, 2014.

[31] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[32] R. Mur-Artal, J. D. Tardós, J. M. M. Montiel, and D. Gálvez-López, "ORB-SLAM2," https://github.com/raulmur/ORB_SLAM2, 2016.

[33] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[34] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry With Fisheye Cameras," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3693–3700, 2018.

[35] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3923–3931.

[36] J. Engel, V. Koltun, and D. Cremers, "DSO: Direct Sparse Odometry," https://github.com/JakobEngel/dso, 2018.

[37] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct Sparse Mapping," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1363–1370, 2020.

[38] J. Zubizarreta, I. Aguinaga, J. D. Tardós, and J. M. M. Montiel, "DSM: Direct Sparse Mapping," https://github.com/jzubizarreta/dsm, 2019.

[39] A. Pumarola, A. Vakhitov, A. Agudo, F. Moreno-Noguer, and A. Sanfeliu, *Relative Localization for Aerial Manipulation with PL-SLAM*. Cham: Springer International Publishing, 2019, pp. 239–248. [Online]. Available: https://doi.org/10.1007/978-3-030-12945-3_17

[40] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM," https://github.com/rubengooj/pl-slam, 2018.

[41] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "PL-VIO," https://github.com/HeYijia/PL-VIO, 2019.

[42] Q. Fu, J. Wang, H. Yu, I. Ali, F. Guo, Y. He, and H. Zhang, "PL-VINS: Real-Time Monocular Visual-Inertial SLAM with Point and Line Features," 2020. [Online]. Available: https://arxiv.org/abs/2009.07462

[43] ——, "PL-VINS: Real-Time Monocular Visual-Inertial SLAM with Point and Line Features," https://github.com/cnqiangfu/PL-VINS, 2020.

[44] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation, year=2007," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572.

[45] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013. [Online]. Available: https://doi.org/10.1177/0278364913481251

[46] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 165–172.

[47] M. K. Paul and S. I. Roumeliotis, "Alternating-Stereo VINS: Observability Analysis and Performance Evaluation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4729–4737.

[48] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization," *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.

[49] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, 02 2014.

[50] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "OKVIS: Open keyframe-based visual-inertial SLAM (ROS version)," https://github.com/ethz-asl/okvis_ros, 2016.

[51] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.

[52] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017. [Online]. Available: https://doi.org/10.1177/0278364917728574

[53] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "ROVIO," https://github.com/ethz-asl/rovio, 2015.

[54] R. Mur-Artal and J. D. Tardós, "Visual-Inertial Monocular SLAM With Map Reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[55] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[56] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," 2019. [Online]. Available: https://arxiv.org/abs/1901.03638

[57] T. Qin, S. Cao, J. Pan, P. Li, and S. Shen, "VINS-Fusion," https://github.com/HKUST-Aerial-Robotics/VINS-Fusion, 2019.

[58] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2510–2517.

[59] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-Inertial Mapping With Non-Linear Factor Recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2020.

[60] V. Usenko and N. Demmel, "BASALT," https://gitlab.com/VladyslavUsenko/basalt, 2019.

[61] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696.

[62] ——, "Kimera," https://github.com/MIT-SPARK/Kimera, 2019.

[63] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," https://github.com/UZ-SLAMLab/ORB_SLAM3, 2020.

[64] I. Alamanos, "ORB-LINE-SLAM," https://github.com/Giannis-Alamanos/ORB-LINE-SLAM, 2022.

[65] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.

[66] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[67] C. Akinlar and C. Topal, "Edlines: Real-time line segment detection by Edge Drawing (ed)," in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 2837–2840.

[68] OpenCV, "Open source computer vision library," 2015.

[69] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[70] L. M. Paz, P. PiniÉs, J. D. TardÓs, and J. Neira, "Large-Scale 6-DOF SLAM With Stereo-in-Hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[71] W. Runzhi, K. Di, W. Wan, and Y. Wang, "Improved Point-Line Feature Based Visual SLAM Method for Indoor Scenes," *Sensors*, vol. 18, p. 3559, 10 2018.

[72] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[73] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. [Online]. Available: https://doi.org/10.1177/0278364915620033

[74] D. Zuñiga-Noël, A. Jaenal, R. Gomez-Ojeda, and J. Gonzalez-Jimenez, "The UMA-VI dataset: Visual–inertial odometry in low-textured and dynamic illumination environments," *The International Journal of Robotics Research*, vol. 39, no. 9, pp. 1052–1060, 2020.

[75] T. K. Kim, "T test as a parametric statistic," *Korean journal of anesthesiology*, vol. 68, no. 6, pp. 540–546, 2015.

[76] M. Lange, F. Schweinfurth, and A. Schilling, "DLD: A Deep Learning Based Line Descriptor for Line Feature Matching," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5910–5915.

[77] M. Lange, C. Raisch, and A. Schilling, "WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching," in *Vision, Modeling, and Visualization*, J. Krüger, M. Niessner, and J. Stückler, Eds. The Eurographics Association, 2020.

[78] C. Qiao, T. Bai, Z. Xiang, Q. Qian, and Y. Bi, "Superline: A Robust Line Segment Feature for Visual SLAM," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5664–5670.

[79] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.

[80] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[81] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping: part II," *IEEE Robotics Automation Magazine*, vol. 13, pp. 108 – 117, 10 2006.

[82] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, "A review of recent developments in Simultaneous Localization and Mapping," in *2011 6th International Conference on Industrial and Information Systems*, 2011, pp. 477–482.

[83] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[84] N. Nain, V. Laxmi, B. Bhadviya, D. B M, and M. Ahmed, "Fast Feature Point Detector," in *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, 2008, pp. 301–306.

[85] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555.

[86] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[87] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.

[88] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[89] C. G. Harris and M. J. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, 1988.

[90] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, 2001, pp. 525–531 vol.1.

[91] S. M. Smith, "A New Class of Corner Finder," in *BMVC*, 1992.

[92] Y. Salaün, R. Marlet, and P. Monasse, "Multiscale line segment detector for robust and accurate SfM," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2000–2005.

[93] D. K. Patel, P. A. Bachani, and N. R. Shah, "Distance Measurement System Using Binocular Stereo Vision Approach," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, vol. 02, December 2013.

[94] G.-q. Wei and S. Ma, "Implicit and Explicit Camera Calibration: Theory and Experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, vol. 16, pp. 469 – 480, 06 1994.

[95] Wikipedia contributors, "Pinhole camera model — Wikipedia, the free encyclopedia," 2022, [Online; accessed 23-October-2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pinhole_camera_model&oldid=1110164612

[96] D. C. Brown, "Close-range camera calibration," *PHOTOGRAMMETRIC ENGINEERING*, vol. 37, no. 8, pp. 855–866, 1971.

[97] W. Faig, "CALIBRATION OF CLOSE-RANGE PHOTOGRAMMETRIC SYSTEMS: MATHEMATICAL FORMULATION," *Photogrammetric Engineering and Remote Sensing*, vol. 41, 1975.

[98] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank, "Camera Self-Calibration: Theory and Experiments," in *ECCV*, 1992.

[99] S. Ganapathy, "Decomposition of transformation matrices for robot vision," *Pattern Recognition Letters*, vol. 2, no. 6, pp. 401–412, 1984.

[100] S. Maybank and O. D. Faugeras, "Theory of self-calibration of a moving camera," *The International Journal of Computer Vision*, vol. 8, no. 2, pp. 123–151, 1992.

[101] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 323–344, 1987.

[102] G.-Q. Wei and S. D. Ma, "A complete two-plane camera calibration method and experimental comparisons," in *Fourth International Conference on Computer Vision, ICCV 1993, Berlin, Germany, 11-14 May, 1993, Proceedings*. IEEE, 1993, pp. 439–446.

[103] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[104] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *The International Journal of Computer Vision*, vol. 4, pp. 127–140, 1990.

[105] D. Liebowitz and A. Zisserman, "Metric rectification for perspective images of planes," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, 1998, pp. 482–488.

[106] R. I. Hartley, "Self-calibration from multiple views with a rotating camera," in *Computer Vision — ECCV '94*, J.-O. Eklundh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 471–478.

[107] G. Stein, "Accurate internal camera calibration using rotation, with analysis of sources of error," in *Proceedings of IEEE International Conference on Computer Vision*, 1995, pp. 230–236.

[108] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.

[109] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[110] J.-Y. Bouguet, "Camera calibration toolbox for matlab." Computational Vision at the California Institute of Technology, 2001.

[111] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, USA: MIT Press, 1993.

[112] S. M. Seitz, "Image-based transformation of viewpoint and scene appearance," Ph.D. dissertation, University of Wisconsin, 1997.

[113] W. R. Esposito and C. A. Floudas, *Gauss–Newton method: Least squares, relation to Newton's method Gauss–Newton Method: Least Squares, Relation to Newton's Method*. Boston, MA: Springer US, 2001, pp. 733–738. [Online]. Available: https://doi.org/10.1007/0-306-48332-7_151

[114] J. J. Moré, "The levenberg-marquardt algorithm: Implementation and theory," in *Numerical Analysis*, G. A. Watson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116.

[115] S. Thrun, "Particle Filters in Robotics," in *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[116] P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[117] R. Andersen, *Modern Methods for Robust Regression*, ser. Quantitative Applications in the Social Sciences. SAGE Publications, 2008.

[118] T. Qin and S. Cao, "A-LOAM," https://github.com/HKUST-Aerial-Robotics/A-LOAM.

[119] H. Wang, C. Wang, C. Chen, and L. Xie, "F-LOAM : Fast LiDAR Odometry and Mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[120] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[121] S. Chen, H. Ma, C. Jiang, B. Zhou, W. Xue, Z. Xiao, and Q. Li, "NDT-LOAM: A Real-Time Lidar Odometry and Mapping With Weighted NDT and LFA," *IEEE Sensors Journal*, vol. 22, no. 4, pp. 3660–3671, 2022.

[122] L. Li, X. Kong, X. Zhao, W. Li, F. Wen, H. Zhang, and Y. Liu, "SA-LOAM: Semantic-aided LiDAR SLAM with Loop Closure," 06 2021.

[123] H. Ye, Y. Chen, and M. Liu, "Tightly Coupled 3D Lidar Inertial Odometry and Mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3144–3150.

[124] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142.

[125] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[126] T. Lupton and S. Sukkarieh, "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[127] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[128] R. Elvira, J. D. Tardós, and J. Montiel, "ORBSLAM-Atlas: a robust and accurate multi-map system," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6253–6259.

[129] D. Galvez-López and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[130] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *2011 International Conference on Computer Vision*, 2011, pp. 2352–2359.

[131] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.