ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ ΦΩΤΟΓΡΑΜΜΕΤΡΙΑΣ

Ανάπτυξη και Εφαρμογή μονοεικονικού συστήματος SLAM

Development and Implementation of a Monocular SLAM system

Σταθοπούλου Βασιλίνα

Αθήνα 2021

Επιβλέπων: Καθηγητής Α. Γεωργόπουλος

Στον παππού μου Μιλτιάδη και τη γιαγιά μου Αγγελική

Abstract

Visual SLAM (Simultaneous Localization and Mapping) has been investigated for many years, in several fields such as robotics, AR, documentation of cultural heritage etc. Significant evolution and achievements have been made, with feature-based techniques becoming robust and accurate. The main advantage of SLAM system is that the process of localization and 3D mapping is done concurrently and recursively without any prior knowledge. Feature-based techniques rely on the extraction of feature points. They search for correspondences by comparing the descriptor of each feature point between images and optimize the solution by minimizing the reprojection error. Therefore, they do not use the whole information of an image and they produce a sparse point cloud.

This thesis focuses on the development and implementation of such a Visual SLAM system, called ORB-SLAM 2, which is presented by Mur-Artal et al. (2017). The work is limited to an offline process because the data used had already been acquired. It was tested on two datasets, describing two different environments, an underwater and an urban scenario, in order to evaluate its performance and localization accuracy.
It also proposes the creation of an incremental visual map which aims to provide a more informative view of the environment to the user, making the produced point cloud graspable. This visual map leverages the feature matching procedure of the ORB-SLAM 2 and the produced keyframe trajectory as well.
In particular, the ORB-SLAM 2 was modified in such way to create a text file containing the consecutive keyframes with their correspondences. The proposed program reads the keyframe trajectory file and stores for each pair of keyframes its matches. In case no matches have been found or are less than 4, the ORB operator takes place. Then, the homographies for each pair are estimated inside a RANSAC scheme with respect to the first image plane (reference). The dimensions of the final “mosaic” are computed, and the images get warped. The mosaic is produced and can be displayed incrementally with two different approaches. The proposed approach was tested on the underwater dataset.
The conclusions mainly focus on the evaluation of the ORB-SLAM 2 system and the visual map. Further improvements are also discussed.

Περίληψη

Τα τελευταία χρόνια, το οπτικό SLAM (Visual SLAM) έχει γνωρίσει σημαντική εξέλιξη σε διάφορους τομείς, όπως είναι η ρομποτική, η επαυξημένη πραγματικότητα, η αποτύπωση μνημείων πολιτιστικής κληρονομιάς, κ.ά. Ένα από τα κυριότερα πλεονεκτήματά του είναι ότι η διαδικασία του υπολογισμού της θέσης και της τρισδιάστατης χαρτογράφησης του άγνωστου περιβάλλοντος γίνεται ταυτόχρονα και αναδρομικά, χωρίς καμία αρχική γνώση. Οι τεχνικές που βασίζονται στην εξαγωγή χαρακτηριστικών σημείων, ψάχνουν για συνταυτίσεις σημείων μεταξύ των εικόνων, συγκρίνοντας τους περιγραφείς τους και βελτιστοποιούν τη λύση ελαχιστοποιώντας το σφάλμα επαναπροβολής. Επομένως, δεν χρησιμοποιούν όλη την πληροφορία της εικόνας και γι' αυτό παράγουν ένα αραιό νέφος σημείων.

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη και εφαρμογή ενός τέτοιου οπτικού συστήματος που λειτουργεί σε πραγματικό χρόνο, το ORB-SLAM 2, που αναπτύχθηκε από τους Mur-Artal et al. (2017). Η εφαρμογή πραγματοποιείται εκτός σύνδεσης, επειδή τα δεδομένα που χρησιμοποιήθηκαν είχαν ήδη αποκτηθεί. Ο αλγόριθμος εφαρμόστηκε σε δύο σετ δεδομένων, που περιγράφουν δύο διαφορετικά περιβάλλοντα, ένα υποθαλάσσιο και ένα αστικό, έτσι ώστε να αξιολογηθεί τόσο η απόδοσή του όσο και η ακρίβεια εντοπισμού της θέσης της κάμερας.
Ακόμα, προτείνεται η δημιουργία ενός αυξητικού οπτικού χάρτη, με σκοπό να προσφέρει μια πιο ολοκληρωμένη εικόνα του εκάστοτε περιβάλλοντος στο χρήστη. Ο συγκεκριμένος χάρτης αξιοποιεί τη διαδικασία της εξαγωγής και συνταύτισης σημείων μεταξύ των εικόνων του ORB-SLAM 2 καθώς και το αρχείο της πορείας της κάμερας.
Πιο συγκεκριμένα, ο ORB-SLAM 2 τροποποιήθηκε έτσι ώστε κατά την εκτέλεσή του να παράγει ένα αρχείο με τα ονόματα διαδοχικών εικόνων-"κλειδιά" και τις συνταυτίσεις τους. Η προτεινόμενη μέθοδος "διαβάζει" το αρχείο της πορείας της κάμερας και αποθηκεύει για κάθε ζευγάρι εικόνων-"κλειδιά" τις συνταυτίσεις του. Σε περίπτωση που αυτές δεν υπάρχουν ή υπάρχουν αλλά είναι λιγότερες από 4, εκτελείται ο ανιχνευτής ORB. Στη συνέχεια, υπολογίζονται οι ομογραφίες για κάθε ζεύγος ως προς το επίπεδο της πρώτης εικόνας (εικόνα αναφοράς) και οι εικόνες μετασχηματίζονται. Υπολογίζονται οι διαστάσεις της τελικής εικόνας και προβάλλονται πάνω σε αυτή οι μετασχηματισμένες εικόνες. Ο οπτικός χάρτης παράγεται και οπτικοποιείται χρησιμοποιώντας δύο μεθόδους. Η μέθοδος εφαρμόστηκε σε υποθαλάσσιο περιβάλλον.
Τα συμπεράσματα επικεντρώνονται κυρίως στην αξιολόγηση του ORB-SLAM 2 και του οπτικού χάρτη. Ακόμη, προτείνονται βελτιώσεις για ενίσχυσή τους.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας Διπλωματικής Εργασίας και των σπουδών μου στο Ε.Μ.Π, θα ήθελα να ευχαριστήσω ιδιαιτέρως τον επιβλέποντα Καθηγητή μου, κο. Ανδρέα Γεωργόπουλο, τόσο για την βοήθεια και τις συμβουλές του κατά τη διάρκεια εκπόνησής της, όσο και για το αμέριστο ενδιαφέρον και την καθοδήγησή του όλα αυτά τα χρόνια, που με έκαναν να ανακαλύψω και να αγαπήσω το αντικείμενο του Τοπογράφου Μηχανικού και ιδιαίτερα της Φωτογραμμετρίας.

Ευχαριστώ τον Παναγιώτη Αγραφιώτη, Διδάκτορα της σχολής Α.Τ.Μ του Ε.Μ.Π για τις ιδέες του για την εκπόνηση της παρούσας Δ.Ε.

Ακόμα, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου για την αδιάκοπη στήριξη και εκπαίδευση που έλαβα χάρη σε αυτούς, τους φίλους μου και ιδιαίτερα τον Θ.Μ για την ενθάρρυνσή του καθόλη τη διάρκεια της παρούσας εργασίας.

I would also like to express my sincere gratitude to Erica Nocerino for the fruitful discussions about the subject and her helpful comments.

List of contents

List of Figures

List of Tables

Abbreviations

AR Augmented Reality

BA Bundle Adjustment

BoW Bag-of-Words

BRIEF Binary Robust Independent Elementary Features

CNN Convolutional Neural Network

CV Computer Vision

DL Deep Learning

EKF Extended Kalman Filter

FAST Features from Accelerated Segment Test

FoV Field of View

GPS Global Positioning System

IMU Inertial Measurement Unit

LIDAR Light Detection And Ranging

ORB Oriented FAST and Rotated BRIEF

PCD Point Cloud Data

PCL Point Cloud Library

RCNN Recurrent Convolutional Neural Network

ReLU Rectified Linear Unit

RGB-D Red Green Blue-Depth

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SfM Structure from Motion

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SURF Speeded Up Robust Features

VO Visual Odometry

1. Εισαγωγή

Με την ολοένα και μεγαλύτερη ανάπτυξη της τεχνολογίας στον κλάδο της Φωτογραμμετρίας, της όρασης υπολογιστών και του αυτοματισμού, πιο περίπλοκες και σύνθετες εργασίες-διαδικασίες μπορούν να οριστούν και να αντιμετωπιστούν από τους ανθρώπους. Τα τελευταία χρόνια, οι γρήγοροι και έντονοι ρυθμοί ζωής δημιουργούν την ανάγκη μιας πιο αυτόματης απόδοσης των πραγμάτων. Ως Φωτογραμμετρία καλείται η επιστήμη που ασχολείται με την εξαγωγή μετρικής πληροφορίας από εικόνες. Η όραση υπολογιστών αποτελεί ένα πεδίο που εστιάζει στην αυτόματη ερμηνεία και κατανόηση του περιεχομένου των ψηφιακών εικόνων και βίντεο. Σήμερα, οι υπολογιστές έχουν την ικανότητα να αναγνωρίζουν και να ταξινομούν αντικείμενα με ακρίβεια και αποτελεσματικότητα. Οι τεχνικές που βασίζονται στην εικόνα, για τη μέτρηση και τρισδιάστατη μοντελοποίηση αντικειμένων έχουν προκαλέσει μεγάλο ενδιαφέρον, καθώς έχουν αναπτυχθεί νέα εργαλεία που αυξάνουν την αυτοματοποίηση των φωτογραμμετρικών διαδικασιών. Η χρήση καμερών προσφέρει χαμηλή κατανάλωση ενέργειας, εμπλουτισμένη οπτική πληροφορία και απλότητα στην εφαρμογή διαφόρων διαδικασιών, συγκριτικά με άλλους αισθητήρες, όπως τα lidar. Το Structure from Motion (Δομή από Κίνηση) αποτελεί την πιο διαδεδομένη τεχνική για τη 3Δ ανακατασκευή αντικειμένων λόγω της αποτελεσματικότητας και του χαμηλού κόστους. Καθορίζει τη χωρική και γεωμετρική σχέση του αντικειμένου με βάση την κίνηση της κάμερας. Υπολογίζει ταυτόχρονα τις παραμέτρους της κάμερας και το 3Δ χάρτη του περιβάλλοντος, συνδυάζοντας 2Δ εικόνες που λαμβάνονται από διαφορετικές οπτικές γωνίες (Karmacharya et al. 2019). Ακόμα, διαχειρίζεται εικόνες που έχουν ληφθεί από διαφορετικές κάμερες και είναι ανοργάνωτες. Ωστόσο, οι τεχνικές 3Δ ανακατασκευής που βασίζονται στις εικόνες απαιτούν μεγάλο χρόνο για την απόκτηση και επεξεργασία των δεδομένων, λόγω της υψηλής υπολογιστικής πολυπλοκότητάς τους (υπολογιστικό κόστος n^4), κάτι που δυσχεραίνει τη λειτουργία σε πραγματικό χρόνο (Wu, 2013). Η ανάπτυξη του οπτικού SLAM (Visual SLAM) τα τελευταία χρόνια καθιστά εφικτή την 3Δ ανακατασκευή σε πραγματικό χρόνο, σε διάφορα περιβάλλοντα, από μικρούς εσωτερικούς έως και μεγάλους εξωτερικούς χώρους, προϋποθέτοντας ότι η βαθμονόμηση έχει πραγματοποιηθεί. Στη σημερινή εποχή, το οπτικό SLAM έχει γνωρίσει αξιοσημείωτη εξέλιξη, παρέχοντας ταχύτητα, ευελιξία, και αποτελώντας σημαντική τεχνολογία για πληθώρα επιστημονικών κλάδων. Αναφέρεται στην τεχνική υπολογισμού της θέσης και του προσανατολισμού ενός αισθητήρα βάσει του περιβάλλοντός του, ενώ παράλληλα, ανακατασκευάζει το 3Δ περιβάλλον γύρω του, θεωρώντας ως περιορισμούς τις εκτιμήσεις της θέσης. Η γεωμετρική σχέση μεταξύ του αισθητήρα και του περιβάλλοντος μπορεί να προσδιοριστεί από τις οργανωμένες ακολουθίες εικόνων. Το οπτικό SLAM χρησιμοποιείται σε πλήθος εφαρμογών στον τομέα της ρομποτικής, της αυτόνομης οδήγησης, της επαυξημένης πραγματικότητας και της επίγειας και ενάλιας πολιτιστικής κληρονομιάς (Association for Advancing Automation, 2018). Μία από τις σημαντικότερες ευκαιρίες του συστήματος SLAM είναι η αντικατάσταση του GPS σε περιβάλλοντα που αυτό υπολειτουργεί ή δε λειτουργεί καθόλου. Παραδείγματα τέτοιων περιβαλλόντων είναι τα μεγάλα αστικά κέντρα, οι εσωτερικοί χώροι, οι υποθαλάσσιες και υπόγειες περιοχές, τα τούνελ, κ.ά. Υπάρχουν διάφορες κατηγορίες SLAM συστημάτων, με τις βασικότερες να είναι οι άμεσες και έμμεσες μέθοδοι. Οι άμεσες μέθοδοι στηρίζονται στις τιμές φωτεινότητας των pixels και εκμεταλλεύονται την πληροφορία ολόκληρης της εικόνας, με αποτέλεσμα να είναι ανθεκτικές και αποτελεσματικές σε περιβάλλοντα χωρίς υφή. Ωστόσο, το υψηλό υπολογιστικό τους κόστος καθώς και η ευαισθησία τους στις αλλαγές φωτεινότητας εμποδίζουν την εφαρμογή τους σε πραγματικό χρόνο και περιορίζουν τη χρήση τους. Αντίθετα, οι έμμεσες μέθοδοι βασίζονται στην εξαγωγή χαρακτηριστικών σημείων και τη συνταύτισή τους, χρησιμοποιώντας πληθώρα ανιχνευτών και περιγραφέων, όπως ο FAST (Rosten et el., 2006), HARRIS (Harris et al. 1988), BRIEF (Calonder, 2010), ORB (Rublee et al., 2011), SIFT (Lowe, 1999), SURF (Bay et al. 2006) κ.ά. Οι αλγόριθμοι που στηρίζονται στα γεωμετρικά σημεία εξάγουν πιο εύκολα τη γεωμετρία από τις εικόνες και επιτρέπουν τη συνταύτιση μεγάλων στερεοβάσεων, χρησιμοποιώντας περιγραφείς που παραμένουν αναλλοίωτοι σε αλλαγές κλίμακας, θέασης και έντασης. Ωστόσο, το νέφος σημείων που παράγουν είναι αραιό και συχνά άχρωμο.

Η εργασία αυτή εστιάζει στην ανάπτυξη και εφαρμογή ενός μονοεικονικού συστήματος οπτικού SLAM. Η εφαρμογή περιορίζεται σε επεξεργασία εκτός σύνδεσης, λόγω του ότι τα δεδομένα που χρησιμοποιήθηκαν είχαν ήδη αποκτηθεί. Το σύστημα SLAM που χρησιμοποιήθηκε είναι το ORB-SLAM 2 και εφαρμόστηκε σε δύο διαφορετικά περιβάλλοντα, ένα υποθαλάσσιο και ένα αστικό. Τα αποτελέσματα που προέκυψαν, σχολιάστηκαν και ύστερα αξιολογήθηκαν, συγκρίνοντάς τα με τα πραγματικά δεδομένα. Επιπροσθέτως, η συγκεκριμένη εργασία ερευνά την παραγωγή ενός οπτικού χάρτη που αποτελείται από ένα υποσύνολο εικόνων (εικόνες-κλειδιά) που καθορίζονται από τον ORB-SLAM 2. Βασίζεται σε δύο βασικές διαδικασίες του αλγορίθμου, την εξαγωγή χαρακτηριστικών σημείων και τη συνταύτισή τους. Με αυτόν τον τρόπο, ο οπτικός χάρτης παρέχει μεγαλύτερη ακρίβεια και δίνει μια πιο "ξεκάθαρη" εικόνα του περιβάλλοντος στο χρήστη.
Το σύστημα δεν έχει τη δυνατότητα να αποθηκεύσει και να φορτώσει τον παραγόμενο χάρτη. Η αποθήκευσή του είναι απαραίτητη για μεταγενέστερη επεξεργασία ενώ η φόρτωσή του συμβάλλει σε μια δεύτερη εκτέλεση του αλγορίθμου. Πιο αναλυτικά, η φόρτωση και ο εντοπισμός της θέσης σε έναν ήδη κατασκευασμένο χάρτη βελτιώνει την τοπική ακρίβεια του αποτελέσματος. Η συγκεκριμένη εργασία στοχεύει στη δημιουργία μιας συνάρτησης αποθήκευσης του 3Δ χάρτη σε ένα αρχείο τύπου PCD, μόλις η διαδικασία ολοκληρωθεί.

2. Επισκόπηση επιστημονικής περιοχής

Το πρόβλημα του SLAM, δηλαδή της ταυτόχρονης απόδοσης του 3Δ χάρτη ενός άγνωστου περιβάλλοντος και της εύρεσης της πορείας του αισθητήρα μέσα σε αυτόν, σε πραγματικό χρόνο, έχει γνωρίσει σημαντική εξέλιξη την τελευταία δεκαετία και αποτελεί σημαντικό κομμάτι στο πεδίο της όρασης υπολογιστών. Παρ' όλα αυτά, είναι μια τεχνολογία που απαιτεί περαιτέρω ανάλυση και έρευνα, λόγω της δυσκολίας και της πολυπλοκότητάς της. Το οπτικό SLAM αναφέρεται στις τεχνικές εκείνες που χρησιμοποιούν ως αισθητήρα την κάμερα. Οι τεχνικές που βασίζονται στη χρήση μόνο μιας κάμερας ονομάζονται οπτικό μονοεικονικό SLAM. Χωρίζεται σε δύο κύριες κατηγορίες, τις έμμεσες που βασίζονται στην εξαγωγή χαρακτηριστικών σημείων και στις άμεσες που βελτιώνονται κατευθείαν από τις τιμές έντασης των pixels. Η πρώτη κατηγορία χωρίζεται σε δύο υποκατηγορίες, η μία στηρίζεται στη χρήση εικόνων-κλειδιά (keyframes), ενώ η δεύτερη σε τεχνικές φιλτραρίσματος που χρησιμοποιούν όλη την πληροφορία της εικόνας.

2.1. Άμεσες μέθοδοι

Η πρώτη μέθοδος που χρησιμοποιήθηκε είναι το DTAM (Dense Tracking and Mapping), από τους Newcombe et al. (2011), η οποία παράγει ένα πυκνό 3Δ χάρτη του περιβάλλοντος και υπολογίζει τη θέση της κάμερας μέσα σε αυτόν. Ο χάρτης δημιουργείται με βάση την τεχνική της τριγωνοποίησης, ενώ η εκτίμηση της θέσης βασίζεται στην σύγκριση της τρέχουσας εικόνας με συνθετικές εικόνες, οι οποίες δημιουργούνται από τον ήδη υπάρχοντα χάρτη. Το πρόβλημα αυτό βελτιστοποιείται μέσω της μείωσης του φωτομετρικού σφάλματος ολόκληρης της εικόνας, η οποία έχει ως αποτέλεσμα την εκτίμηση του βάθους του κάθε pixel της. Η μέθοδος αυτή είναι ανθεκτική σε φαινόμενα θορύβου αλλά απαιτεί ισχυρή κάρτα γραφικών προκειμένου να πραγματοποιηθεί σε πραγματικό χρόνο.

Το LSD-SLAM (Large scale Direct monocular SLAM) που αναπτύχθηκε από τους Engel et al. (2014) αποτελεί βελτίωση της μεθόδου του DTAM με βασική διαφορά ότι το πρώτο δεν χρησιμοποιεί ολόκληρη την εικόνα, αλλά μόνο εκείνα τα pixels που παρουσιάζουν υψηλές αλλαγές χρώματος με τα γειτονικά τους. Αυτό έχει ως αποτέλεσμα την παραγωγή ενός μέτρια πυκνού νέφους σημείων. Κατά την αρχικοποίηση του αλγορίθμου, οι τιμές βάθους της πρώτης εικόνας ορίζονται τυχαία για κάθε pixel. Στη συνέχεια, προκειμένου να υπολογιστούν σωστά, πρέπει να γίνει σημαντική μετάθεση της κάμερας. Η διαδικασία της χαρτογράφησης και του υπολογισμού της θέσης πραγματοποιείται με παρόμοιο τρόπο με το DTAM. Ακόμα, περιλαμβάνει μέθοδο κλεισίματος βρόγχου για να μειώσει το συνολικό σφάλμα που προκύπτει. Ωστόσο, η μέθοδος αυτή είναι επιρρεπής στο σφάλμα κλίμακας λόγω της άγνοιας του βάθους

και γι' αυτό χρησιμοποιείται μία τεχνική συσχέτισης έτσι ώστε εικόνες με διαφορετική κλίμακα να μπορούν να συσχετιστούν.

2.2. Έμμεσες μέθοδοι

Μια αντιπροσωπευτική και επαναστατική μέθοδος που στηρίζεται στην χρήση εικόνων-κλειδιών είναι το PTAM (Parallel Tracking and Mapping), που παρουσιάστηκε από τους Klein et al. (2007). Διαχώρισαν το πρόβλημα του SLAM σε δύο επιμέρους διαδικασίες που μπορούν να εκτελεστούν παράλληλα, το tracking που αποτελεί την εκτίμηση της θέσης της κάμερας και το mapping, που συνίσταται στη χαρτογράφηση του περιβάλλοντος. Τα βήματα του αλγορίθμου μπορούν να συνοψιστούν ως εξής: (i) Για την αρχικοποίηση του συστήματος χρησιμοποιείται ο 5-point αλγόριθμος του Nister et al. (2004) για ένα στερεοζεύγος, ο οποίος υπολογίζει τον δεσμευμένο επιπολικό πίνακα. Έτσι προκύπτει το διάνυσμα μετάθεσης και στροφής του στερεοζεύγους, από το οποίο μπορεί να κατασκευαστεί ο αρχικός χάρτης. (ii) Μέσω της επιπολικής γεωμετρίας, δημιουργούνται καινούργια σημεία, ενισχύοντας το χάρτη. (iii) Η διαδικασία του tracking και του mapping πραγματοποιούνται παράλληλα έτσι ώστε το πρώτο να μην επηρεαστεί από το υπολογιστικό κόστος του δεύτερου. Μια σημαντική συνεισφορά του PTAM είναι το relocalization, το οποίο πραγματοποιείται όταν η θέση της τρέχουσας κάμερας δεν μπορεί να υπολογιστεί με βάση το χάρτη. Εφόσον, η μέθοδος αυτή στηρίζεται στην εξαγωγή χαρακτηριστικών σημείων της εικόνας, δημιουργεί ένα αραιό αλλά ακριβές νέφος σημείων.

Μία γνωστή μέθοδος φιλτραρίσματος αποτελεί το 1-point RANSAC for EKF Filtering που αναπτύχθηκε από τους Civera et al. (2010). Ο 1-point RANSAC χρησιμοποιεί ένα ζεύγος αντίστοιχων σημείων για την εκτίμηση της κίνησης, οδηγώντας στη μικρότερη παραμετροποίηση του μοντέλου. Στη μέθοδο Camera-centered EKF οι θέσεις των χαρακτηριστικών σημείων καθώς και της κάμερας αναφέρονται σε μια εικόνα που είναι γειτονική με την τρέχουσα και όχι σε ένα παγκόσμιο πλαίσιο αναφοράς. Με αυτόν τον τρόπο, τα σφάλματα μειώνονται, οδηγώντας σε μικρότερες αβεβαιότητες. Εκμεταλλεύεται την πιθανότητα πρόβλεψης που λαμβάνεται από το EKF (Extended Kalman Filter), με σκοπό να μειώσει το υπολογιστικό κόστος της διαδικασίας που απορρίπτει τις λάθος αντιστοιχίες σημείων. Τα βασικά βήματά του είναι τα εξής: (1) Διαλέγει μια εικόνα και ξεκινάει την επεξεργασία της εξάγοντας χαρακτηριστικά σημεία. (2) Τοποθετεί τα σημεία αυτά στη δεύτερη εικόνα στην αντίστοιχη θέση. (3) Δημιουργεί έναν κύκλο γύρω από το καθένα θέτοντας μια τιμή κατωφλίου για την ακτίνα και ψάχνει μέσα σε αυτόν τα ίδιο σημείο. (4) Ζωγραφίζει τους κύκλους ανάλογα με το αν τα σημεία βρίσκονται εντός ή εκτός από αυτούς και σταματάει τη διαδικασία αν το σημεία δεν βρίσκονται στην εικόνα. Γι' αυτό, οι εικόνες χρειάζεται να έχουν μεγάλη επικάλυψη μεταξύ τους.

2.3. Συστήματα οπτικού SLAM που βασίζονται σε τεχνικές βαθιάς μάθησης

Το DF-SLAM (Deep Features based SLAM) αναπτύχθηκε από τους Kang et al. (2019) και είναι μια μέθοδος οπτικού SLAM που βασίζεται στην εξαγωγή περιγραφέων βαθιών χαρακτηριστικών σημείων που αποκτούνται από ένα νευρωνικό δίκτυο. Οι υπόλοιπες διαδικασίες παραμένουν ίδιες με τις συμβατικές μεθόδους, προκειμένου να επιτευχθεί η λειτουργία σε πραγματικό χρόνο αλλά και διατηρηθεί η αποτελεσματικότητα των χαρακτηριστικών σημείων

Οι Wang et al. (2017) πρότειναν τη δημιουργία ενός επαναλαμβανόμενου νευρωνικού δικτύου (Recurrent Convolutional Neural Network) που βασίζεται στη σημαντικότητα των γεωμετρικών χαρακτηριστικών σημείων και εκμεταλλεύεται την πληροφορία μεταξύ διαδοχικών εικόνων προκειμένου να εξάγει την μεταξύ τους κίνηση. Το σύστημα αυτό συνίσταται από ένα συμβατικό νευρωνικό δίκτυο CNN και ένα επαναλαμβανόμενο RNN. Αρχικά, το ζεύγος εικόνων εισέρχεται στο CNN όπου μειώνονται οι διαστάσεις του και προκύπτουν τα "μαθημένα" χαρακτηριστικά σημεία. Αυτά εισάγονται στο RNN για να

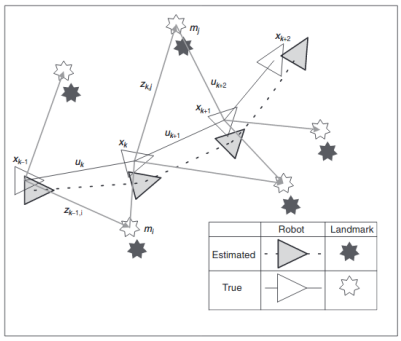
προκύψει η σχετική τους κίνηση. Επειδή το RNN έχει την ικανότητα να κρατάει στη μνήμη του "κρυφές" καταστάσεις και να πραγματοποιεί λούπες, το προτεινόμενο σύστημα μπορεί να διαχειριστεί μεγάλες διαδρομές. Ένα πολύ σημαντικό χαρακτηριστικό του είναι ο υπολογισμός την απόλυτης κλίμακας χωρίς καμία αρχική γνώση.

2.4. Εφαρμογές οπτικού SLAM/Odometry

Η διπλωματική εργασία της Κουρουνιώτη (2018) χωρίζεται σε δύο τμήματα. Το πρώτο κομμάτι έχει σκοπό την παραγωγή και αξιολόγηση ορθοφωτογραφιών με δύο διαφορετικούς τρόπους ενώ το δεύτερο στηρίζεται στην ανάλυση του αλγορίθμου "1-point RANSAC for EKF Filtering" για σκοπούς γρήγορης αποτύπωσης. Αρχικά, ένας σαρωτής laser (ZEB-REVO) που βασίζεται στην τεχνική του SLAM σαρώνει την περιοχή και παράγει το άχρωμο αλλά πυκνό νέφος σημείων. Στη συνέχεια, αυτό χρωματίζεται με βάση ένα δεύτερο νέφος σημείων που δημιουργείται από εικόνες που έχουν παρθεί από τρεις διαφορετικές κάμερες και σε διαφορετικά ύψη. Ακόμα, παράγεται ένα τρίτο έγχρωμο νέφος σημείων από μία κάμερα DSLR και το οποίο αποτελεί νέφος αναφοράς. Ύστερα από τις απαραίτητες διαδικασίες παράγονται οι ορθοφωτογραφίες και συγκρίνονται. Στο δεύτερο τμήμα της διπλωματικής εκτελείται ο αλγόριθμος της οδομετρίας σε τρία διαφορετικά περιβάλλοντα, έναν εσωτερικό διάδρομο και δύο στενούς εξωτερικούς δρόμους. Μερικά από τα συμπεράσματα που εξήχθησαν είναι ότι οι εικόνες χρειάζονται σημαντική επικάλυψη προκειμένου ο αλγόριθμος να βρίσκει σημεία και να μην τερματίζει τη διαδικασία, η απότομη και γρήγορη κίνηση της κάμερας καθώς και οι επιφάνειες χωρίς υφή δυσκολεύουν την εξαγωγή χαρακτηριστικών σημείων, ο αλγόριθμος επηρεάζεται από τον τύπο φακού και τέλος, οι μετρήσεις είναι εξαρτώμενες μεταξύ τους καθώς σημεία που υπολογίστηκαν σε λάθος θέση, θα εξακολουθούν να εμφανίζονται λάθος λόγω του ότι η εκτίμηση της θέσης της κάμερας στηρίζεται στην προηγούμενη θέση.

Οι Ortiz-Coder et al. (2019) ανέπτυξαν ένα φορητό σύστημα χαρτογράφησης αποτελούμενο από δύο κάμερες A και B, με σκοπό τη 3Δ ανακατασκευή ενός αρχαιολογικού χώρου στην πόλη της Ισπανίας, το "Ρωμαϊκό Υδραγωγείο των θαυμάτων". Στόχος τους είναι η δημιουργία ενός πλήρους αυτόνομου συστήματος με συνεχή λήψη δεδομένων χωρίς καμιά ανθρώπινη παρέμβαση. Τα βασικά βήματα είναι: (1) Οι δύο κάμερες ξεκινούν την καταγραφή βίντεο με 25 και 4 fps, αντίστοιχα. Ο αλγόριθμος ORB-SLAM (Mur-Artal et al. 2015) ενσωματώνεται στο σύστημα, υπολογίζοντας τις θέσεις των δύο καμερών, αφού πρώτα πραγματοποιηθεί η βαθμονόμηση. (2) Στις εικόνες που λήφθηκαν από την κάμερα B εφαρμόζονται κάποια φίλτρα με στόχο την απόρριψη περιττών εικόνων βάσει κάποιων κριτηρίων όπως, η μικρή στερεοβάση. (3) Στη συνέχεια, πραγματοποιείται μια διαδικασία τμηματοποίησης, ομαδοποιώντας τρεις συνεχόμενες εικόνες που "μοιράζουν" αρκετά κοινά χαρακτηριστικά σημεία. (4) Το έγχρωμο νέφος σημείων παράγεται χρησιμοποιώντας ένα φωτογραμμετρικό λογισμικό. Στο τέλος, κατέληξαν στο συμπέρασμα ότι το σύστημά τους μπορεί να ανταγωνιστεί τις κλασικές φωτογραμμετρικές διαδικασίες όσον αφορά την ακρίβεια και το χρόνο.

3. Θεωρητική προσέγγιση



Εικόνα 1. Το πρόβλημα του SLAM.

Η παραπάνω εικόνα περιγράφει το βασικό πρόβλημα των συστημάτων SLAM όπου ένα ρομπότ κινείται σε ένα άγνωστο περιβάλλον (λευκά βέλη). Ενώ κινείται, έχει αισθητήρες για να μπορεί να αντιληφθεί τα ορόσημα (λευκά) στο περιβάλλον σε σχέση με τη θέση του. Στην πραγματικότητα, αυτές οι παρατηρήσεις καθώς και οι κινήσεις του ρομπότ περιλαμβάνουν θόρυβο, οδηγώντας σε αβεβαιότητα. Έτσι, το σύστημα θεωρεί ότι κινείται σύμφωνα με τα γκρι βέλη με βάση π.χ. εντολές που του δίνονται. Με αυτόν τον τρόπο πλέον, παρατηρεί τα μαύρα ορόσημα έναντι των πραγματικών. Ο πραγματικός χάρτης και ο χάρτης που παράγεται από το SLAM διαφέρουν, όπως και η πορεία του ρομπότ, αφού στηρίζεται στη θέση των ορόσημων. Η μετακίνηση μεταξύ των μαύρων και λευκών ορόσημων υποδηλώνει το σφάλμα που υπεισέρχεται.

Σε γενικές γραμμές, το SLAM είναι μια διαδικασία κατά την οποία ένα αυτοκινούμενο ρομπότ αναδημιουργεί το 3Δ χάρτη του άγνωστου περιβάλλοντός του, ενώ παράλληλα εκτιμά τη θέση του, χρησιμοποιώντας αυτόν το χάρτη. Τα τελευταία οπτικά συστήματα αποτελούνται από τρία βασικά στάδια, την αρχικοποίηση (initialization), την παρακολούθηση (tracking) και τη χαρτογράφηση (mapping). Μετά τον ορισμό ενός συστήματος αναφοράς, τμήμα του περιβάλλοντος αρχίζει να κατασκευάζεται, γνωστό ως “αρχικός χάρτης”. Αμέσως μετά, ο χάρτης αυτός χρησιμοποιείται για τον υπολογισμό της θέσης της τρέχουσας κάμερας, λύνοντας ένα PnP problem (Appendix). Όταν η κάμερα συλλαμβάνει περιοχές που δεν έχουν ανακαλυφθεί ξανά, ο χάρτης επεκτείνεται μέσω της διαδικασίας της χαρτογράφησης. Τα σύγχρονα συστήματα εισάγουν δύο επιπλέον διαδικασίες, το relocalization (όταν η θέση της τρέχουσας εικόνας δεν μπορεί να υπολογιστεί με βάση τα σημεία του χάρτη) και το κλείσιμο του βρόγχου (loop closing). Οι διαδικασίες αυτές είναι πολύ σημαντικές καθώς επιτρέπουν λειτουργίες μεγάλης διάρκειας. Το relocalization βοηθά το σύστημα να ανακάμψει αυτόματα από την αποτυχία του tracking, που μπορεί να προκληθεί από θόλωμα, από απότομες και ακανόνιστες κινήσεις της κάμερας καθώς και αποκρύψεις. Η διαδικασία εντοπισμού του βρόγχου ενισχύει το σύστημα SLAM με τέτοιο τρόπο ώστε να μπορεί να αναγνωρίζει περιοχές από τις οποίες έχει ξαναπεράσει και επομένως να διορθώνει το σφάλμα που προκύπτει. Το κλασικό πρόβλημα που ανακύπτει είναι ότι δύο περιοχές στο χάρτη αποτελούν το ίδιο μέρος στην πραγματικότητα, με αποτέλεσμα να δημιουργούνται διπλά σημεία. Επομένως, το σύστημα πρέπει να υπολογίσει ένα μετασχηματισμό, για να “ενώσει” τα δύο κομμάτια, δηλαδή να κλείσει το βρόγχο. Γενικά, οι τεχνικές SLAM επικεντρώνονται στην ελαχιστοποίηση του σφάλματος επαναπροβολής των σημείων, συνήθως μέσω της συνόρθωσης της δέσμης (Bundle Adjustment). Ο κύριος σκοπός του οπτικού SLAM είναι η λειτουργία σε πραγματικό χρόνο. Γι’ αυτό το λόγο, στις περισσότερες περιπτώσεις, η βελτιστοποίηση τόσο της θέσης του αισθητήρα όσο και των σημείων του χάρτη πραγματοποιείται χωριστά αλλά ταυτόχρονα, πριν τελικά συγχωνευτούν.

Περιγραφή συστήματος ORB-SLAM 2

Γιατί ORB-SLAM 2?

Τα τελευταία χρόνια, η πλειονότητα των οπτικών συστημάτων SLAM κατασκευάζουν χάρτες που χρησιμοποιούνται μόνο για τον υπολογισμό της θέσης της κάμερας κατά τη διάρκεια μιας συγκεκριμένης λειτουργίας. Επομένως, δεν μπορούν να υπολογίσουν τη θέση μιας διαφορετικής κάμερας ή της ίδιας από διαφορετικές οπτικές γωνίες. Ακόμα, δεν μπορούν να ανταπεξέλθουν σε δυναμικά περιβάλλοντα. Γι' αυτό, οι Mur-Artal et al. (2017) ανέπτυξαν ένα σύστημα οπτικού SLAM που βασίζεται σε γεωμετρικά χαρακτηριστικά σημεία, τόσο για μονοεικονικές όσο και για στερεοσκοπικές και RGB-D λειτουργίες, το ORB-SLAM 2. Λειτουργεί σε πραγματικό χρόνο και μπορεί να διαχειρίζεται πληθώρα διαφορετικών περιβαλλόντων, από μικρού και μεγάλου μεγέθους έως εσωτερικά και εξωτερικά, αντιμετωπίζοντας βιομηχανικές αλληλουχίες, αλληλουχίες από drones κ.ά. Ο αλγόριθμος χρησιμοποιεί τα ίδια σημεία για όλες τις διαδικασίες του μειώνοντας το υπολογιστικό κόστος και επιτρέποντας μια αξιόπιστη και αποτελεσματική διαδικασία. Χρησιμοποιεί ORB χαρακτηριστικά σημεία (ανεπηρέαστα από αλλαγές φωτεινότητας και γωνιών θέασης) που οδηγούν σε εκτέλεσή του σε πραγματικό χρόνο, χωρίς την απαίτηση GPUs. Αυτό επιτρέπει τη συνταύτιση σημείων μεγάλων βάσεων, ενισχύοντας τη διαδικασία της συνόρθωσης δέσμης. Τα βασικά στάδια του αλγορίθμου είναι η αρχικοποίηση, η παρακολούθηση, η χαρτογράφηση, το κλείσιμο του βρόγχου και η συνολική συνόρθωση της δέσμης.

A. Initialization (Αρχικοποίηση)

Σκοπός της διαδικασίας αυτής είναι ο υπολογισμός της σχετικής θέσης δύο εικόνων, ώστε να είναι δυνατή η τριγωνοποίηση των κοινών τους σημείων, προκειμένου δημιουργηθεί ο αρχικός χάρτης για επίπεδες και μη επίπεδες σκηνές. Είναι μια πλήρως αυτόματη διαδικασία που υπολογίζει δύο γεωμετρικά μοντέλα, μια ομογραφία κατάλληλη για επίπεδα αντικείμενα και έναν επιπολικό πίνακα (fundamental matrεπιποix) για μη επίπεδα. Πρέπει να σημειωθεί ότι η διαδικασία απαιτεί την ύπαρξη σημαντικής παράλλαξης. Ανάλογα με το ποια μέθοδος επιστρέφει καλύτερα αποτελέσματα, γίνεται η επιλογή του κατάλληλου γεωμετρικού μοντέλου και ο χάρτης αρχικοποιείται. Αν δεν βγει κάποιο σαφές αποτέλεσμα, η διαδικασία ξεκινάει από την αρχή. Στο τέλος, ο παραγόμενος χάρτης βελτιστοποιείται μέσω της συνόρθωσης της δέσμης. Το σύστημα αναφοράς ορίζεται ως εξής:

Αρχή: η πρώτη προς επεξεργασία εικόνα
Μονάδες: η σχετική θέση των δύο πρώτων εικόνων
Κλίμακα: η διάμεσος τιμή βάθους του αρχικού χάρτη

B. Tracking (Παρακολούθηση)

Μόλις ένας αρχικός χάρτης έχει δημιουργηθεί, πραγματοποιείται το tracking που υπολογίζει τη θέση της κάθε εισερχόμενης εικόνας ξεχωριστά, ως προς την πρώτη. Αποτελείται από τα παρακάτω στάδια:
- Εξαγωγή χαρακτηριστικών σημείων (FAST corners) και των περιγραφέων τους μέσω του ORB descriptor.
- Εκτίμηση μιας αρχικής θέσης της κάμερας μέσω ενός μοντέλου πρόβλεψης κίνησης. Συγκεκριμένα, ο αλγόριθμος, για ORB σημεία της προηγούμενης εικόνας που αντιστοιχούν σε σημεία του χάρτη αναζητά αντιστοιχίσεις στην τρέχουσα εικόνα σε μια μικρή περιοχή γύρω από την προηγούμενη τους θέση. Επομένως, έχοντας 3Δ-2Δ αντιστοιχίες, υπολογίζεται η θέση, μέσω του PnP προβλήματος χρησιμοποιώντας την επαναληπτική διαδικασία του RANSAC.
- Στην περίπτωση που το προηγούμενο βήμα δεν είναι επιτυχές, η διαδικασία του relocalization ξεκινάει. Υπολογίζεται μια bag-of-words αναπαράσταση της εικόνας και στη συνέχεια, αναζητείται σε μια βάση δεδομένων προηγούμενων εικόνων, η εικόνα-κλειδί (keyframe) με τη

μεγαλύτερη ομοιότητα. Μόλις βρεθεί, υπολογίζεται η θέση της κάμερας βάσει του αλγορίθμου PnP., με τον ίδιο τρόπο που περιγράφτηκε προηγουμένως.
- Αφού υπολογιστεί η θέση της, βελτιστοποιείται μέσω του motion-only Bundle Adjustment (Appendix) όπου ένας τοπικός χάρτης προβάλλεται στην τρέχουσα εικόνα και ψάχνονται συνταυτίσεις. Ο χάρτης αυτός αποτελείται από ένα σύνολο εικόνων-κλειδιών που μοιράζουν τα περισσότερα σημεία με την εικόνα και τις γειτονικές αυτών εικόνες στο γράφημα ορατότητας (Covisibility graph). Στο πρώτο σύνολο περιλαμβάνεται μια εικόνα αναφοράς που περιλαμβάνει τα περισσότερα κοινά σημεία με την τρέχουσα εικόνα. Το Covisibility graph αναπαριστά τη σχέση μεταξύ εικόνων-κλειδιών. Αποτελείται από κόμβους και ακμές. Οι κόμβοι αντιπροσωπεύουν τις εικόνες-κλειδιά και οι ακμές μεταξύ αυτών υπάρχουν όταν οι εικόνες περιέχουν κοινά σημεία χάρτη (τουλάχιστον 15). Στην βελτιστοποίηση δεν λαμβάνουν μέρος τα σημεία τους χάρτη που βρίσκονται στη τρέχουσα εικόνα, διότι φιλτράρονται με βάση τη γωνία θέασης, και την απόσταση της επαναπροβολής τους από το κέντρο της εικόνας.
- Ορισμός της τρέχουσας εικόνας ως εικόνα-κλειδί ή όχι. Γενικά, το σύστημα εξάγει όσες περισσότερες εικόνες-κλειδιά μπορεί και στη συνέχεια, απορρίπτει εκείνες που προσφέρουν ελάχιστη πληροφορία για το περιβάλλον. Συγκεκριμένα, μια εικόνα ορίζεται ως εικόνα-κλειδί όταν, έχουν περάσει τουλάχιστον 20 εικόνες από το relocalization και από την τελευταία είσοδο μιας εικόνας-κλειδί, όταν περιέχει τουλάχιστον 50 χαρακτηριστικά σημεία και όταν "παρακολουθεί" λιγότερο από το 90% των σημείων χάρτη από την εικόνα αναφοράς.

Γ. Local Mapping (Χαρτογράφηση)

Η διαδικασία αυτή στηρίζεται στην επεξεργασία εικόνων-κλειδιά που εισάγονται από το tracking και αποθηκεύονται σε μια "ουρά". Το local mapping ξεκινά την επεξεργασία της παλαιότερης στη σειρά εικόνας-κλειδί ακολουθώντας τα εξής βήματα:
- Με την εισαγωγή μιας εικόνας-κλειδί, το Covisibility graph ενημερώνεται, προσθέτοντας έναν επιπλέον κόμβο και ενημερώνοντας τις ακμές μεταξύ της τρέχουσας και άλλων. Στη συνέχεια, υπολογίζεται μια bag-of-words αναπαράσταση της εικόνας που θα βοηθήσει στο συσχετισμό των δεδομένων σε διάφορα στάδια του αλγορίθμου.
- Αφαίρεση ορισμένων σημείων του χάρτη. Τα σημεία προκειμένου να διατηρηθούν, πρέπει να είναι ορατά από τουλάχιστον το 25% των εικόνων-κλειδιά από τα οποία προβλέπεται να είναι και να φαίνονται σε τουλάχιστον τρεις εικόνες-κλειδιά, εάν έχει περάσει τουλάχιστον ένα μετά τη δημιουργία τους. Η μόνη περίπτωση που κάποιο σημείο μπορεί να διαγραφεί εφόσον πληροί τα παραπάνω κριτήρια είναι όταν εντοπίζεται σε λιγότερες από τρεις εικόνες-κλειδιά. Αυτό μπορεί να συμβεί όταν περιττές εικόνες-κλειδιά απορρίπτονται ή κατά τη διάρκεια της συνόρθωσης δέσμης, όπου αφαιρείται ο θόρυβος.
- Ο χάρτης επεκτείνεται με την τριγωνοποίηση νέων σημείων. Πιο συγκεκριμένα, για κάθε ORB σημείο της τρέχουσας εικόνας-κλειδί που δεν έχει αντιστοιχηθεί, το σύστημα ψάχνει άλλα ORB σημεία σε άλλες εικόνες-κλειδιά μέσω της bag-of-words αναπαράστασης. Όλες οι συνταυτίσεις πρέπει να πληρούν την επιπολική γεωμετρία, αλλιώς απορρίπτονται. Για να ενσωματωθεί ένα σημείο στο χάρτη ελέγχεται αν πληροί ορισμένα κριτήρια όπως είναι, το χαμηλό σφάλμα επαναπροβολής, η ικανοποιητική παράλλαξη, η συνέπεια στην κλίμακα και το θετικό βάθος μπροστά από την κάμερα.
- Η θέση της τρέχουσας εικόνας-κλειδί, των γειτονικών της καθώς και όλων των σημείων του χάρτη που παρατηρούνται από αυτές βελτιστοποιούνται μέσω της τοπικής συνόρθωσης δέσμης (local Bundle Adjustment, Appendix). Εκείνες οι εικόνες-κλειδιά που δεν είναι συνδεδεμένες με την τρέχουσα αλλά μοιράζονται κοινά σημεία του χάρτη, παραμένουν σταθερές.
- Τα keyframes όπου το 90% των σημείων του χάρτη που περιλαμβάνουν παρατηρούνται από τρεις τουλάχιστον εικόνες-κλειδιά στην ίδια ή σε καλύτερη κλίμακα, απορρίπτονται. Με αυτόν τον τρόπο, μειώνεται το υπολογιστικό κόστος της συνόρθωσης και επιτρέπονται λειτουργίες μεγάλης διάρκειας.
Γενικά, τα σημεία του χάρτη (map points) αποθηκεύουν τις 3Δ συντεταγμένες στο χώρο, το κανονικοποιημένο διάνυσμα που είναι το μέσο διάνυσμα των οπτικών ακτινών του, έναν αντιπροσωπευτικό περιγραφέα καθώς και την ελάχιστη και μέγιστη απόσταση από όπου το σημείο αυτό μπορεί να φανεί, με βάση την κλίμακα και την απόσταση από την οποία

παρατηρήθηκε. Οι εικόνες-κλειδιά αποθηκεύουν τη θέση τους στο χώρο, τις παραμέτρους του εσωτερικού προσανατολισμού και τα σημεία που εξήχθησαν είτε ανήκουν στο χάρτη είτε όχι.

Δ. Loop closing (Κλείσιμο βρόγχου)

Η διαδικασία αυτή αποτελεί ένα από τα πιο σημαντικά κομμάτια του συστήματος SLAM. Μέσω της επίσκεψης περιοχών που υπάρχουν ήδη στο χάρτη, είναι δυνατόν να διορθωθεί το σφάλμα που έχει συσσωρευτεί κατά τη διάρκεια της πορείας της κάμερας. Ο αλγόριθμος εντοπίζει το βρόγχο, μέσω μιας διαδικασίας αναγνώρισης τοποθεσίας (place recognition), η οποία αποτελείται από τα εξής στάδια:
- Υπολογίζεται ένας μετασχηματισμός ομοιότητας ανάμεσα στο bag-of-words διάνυσμα της τρέχουσας εικόνας-κλειδί και των γειτονικών της με βάση το Covisibility graph. Το χαμηλότερο σκορ διατηρείται (s_{min}). Στη συνέχεια, το σύστημα ρωτά τη βάση δεδομένων σύμφωνα με τη διαδικασία αναγνώρισης για πιθανές εικόνες-κλειδιά. Εκείνες οι οποίες έχουν χαμηλότερο σκορ από το (s_{min}) και είναι κατευθείαν συνδεδεμένες με την τρέχουσα, απορρίπτονται.
- Για να κλείσει επιτυχώς ένας βρόγχος, πρέπει να υπολογιστεί ένα μετασχηματισμός ομοιότητας μεταξύ της τρέχουσας εικόνας-κλειδί και όλων των πιθανών εικόνων-κλειδιά. Ο μετασχηματισμός με τις περισσότερες αντιστοιχίσεις σημείων που τον ικανοποιούν διατηρείται. Μια συνταύτιση θεωρείται σωστή όταν το άθροισμα των επαναπροβαλλόμενων σφαλμάτων στην τρέχουσα και στις πιθανές εικόνες-κλειδιά είναι λιγότερο από 6 pixels. Ο μετασχηματισμός βελτιστοποιείται ελαχιστοποιώντας το σφάλμα επαναπροβολής των δύο εικόνων-κλειδιά μέσω του RANSAC.
- Μόλις, ο βρόγχος ανιχνευτεί πρέπει να διορθωθεί. Αυτό συνίσταται στη συγχώνευση διπλών σημείων του χάρτη και την ενημέρωση και εισαγωγή νέων άκρων στο Covisibility graph. Πριν κλείσει ο βρόγχος, το σύστημα ενημερώνει τη διαδικασία χαρτογράφησης να σταματήσει την τρέχουσα λειτουργία της. Αυτό το βήμα έχει τεράστια σημασία για την συνοχή του παραγόμενου χάρτη, καθώς και οι δύο αυτές διαδικασίες αλλάζουν τη θέση τόσο της κάμερας όσο και των σημείων του χάρτη. Ο μετασχηματισμός ομοιότητας εφαρμόζεται στη θέση της τρέχουσας εικόνας-κλειδί και των γειτονικών της έτσι ώστε οι πλευρές του βρόγχου να ενωθούν. Όλα τα σημεία του χάρτη που παρατηρούνται από την πιθανή εικόνα-κλειδί και τις γειτονικές της προβάλλονται στην τρέχουσα και στις γειτονικές της εικόνες-κλειδιά. Τα σημεία αυτά αλλά και τα σημεία που επιβεβαίωναν τον υπολογισμό του μετασχηματισμού ομοιότητας ενώνονται.
- Στη συνέχεια, πραγματοποιείται μια συνολική βελτιστοποίηση, ώστε να "κλείσει" ο βρόγχος και να διορθωθεί το συσσωρευμένο σφάλμα. Η βελτιστοποίηση στηρίζεται σε ένα γράφημα το οποίο διανέμει το σφάλμα κατά μήκος του. Συνίσταται σε μετασχηματισμούς ομοιότητας ώστε να διορθωθεί το σφάλμα κλίμακας. Αρχικά βελτιστοποιούνται οι θέσεις της κάμερας και στη συνέχεια, τα σημεία του χάρτη με βάση το μετασχηματισμό που "επιβλήθηκε" στην εικόνα-κλειδί που το περιέχει. Το γράφημα ονομάζεται essential graph και αποτελεί τμήμα του Covisibility graph, δηλαδή περιέχει όλους τους κόμβους αλλά μόνο ένα υποσύνολο των ακμών. Ακόμα, περιλαμβάνει ακμές βρόγχου.

Τέλος, πραγματοποιείται μια συνολική συνόρθωση δέσμης σε ένα ξεχωριστό επεξεργαστικό νήμα λόγω του μεγάλου υπολογιστικού κόστους. Αυτό επιτρέπει την ταυτόχρονη εκτέλεση της χαρτογράφησης και του κλεισίματος βρόγχου. Εάν ένας βρόγχος ανιχνευτεί ενώ τρέχει η διαδικασία της συνόρθωσης, το σύστημα σταματάει τη διαδικασία αυτή και προβαίνει στη διόρθωσή του, που θα "ενεργοποιήσει" ξανά τη συνόρθωση. Μετά την ολοκλήρωσή της, οι βελτιστοποιημένες εικόνες-κλειδιά και τα σημεία του χάρτη πρέπει να ενωθούν με εκείνα που δεν ενημερώθηκαν και εισήχθησαν κατά της διάρκεια της βελτιστοποίησης. Αυτό επιτυγχάνεται με τη "διάδοση" της διόρθωσης των ενημερωμένων εικόνων-κλειδιά στα μη ενημερωμένα και στα σημεία του χάρτη. Οι Mur-Artal et al. (2017) μετά από πειράματα κατέληξαν ότι η συνολική συνόρθωση (global BA) βελτιώνει ελάχιστα την λύση, καθώς το αποτέλεσμα από τη βελτιστοποίηση του γραφήματος είναι πολύ ακριβές.

4. Ανάπτυξη και Εφαρμογή αλγορίθμου και οπτικού χάρτη

4.1. Ανάπτυξη ORB-SLAM 2

4.1.1. Συνάρτηση αποθήκευσης 3Δ χάρτη

Κατά την εφαρμογή του συστήματος, ανοίγει ένα παράθυρο που παρουσιάζει τη δημιουργία του αραιού νέφους σημείων καθώς και την πορεία της κάμερας. Αμέσως μόλις τελειώσει η διαδικασία, το παράθυρο κλείνει και μόνο το αρχείο με την πορεία της κάμερας αποθηκεύεται. Για το λόγο αυτό, κρίθηκε απαραίτητη η αποθήκευση του 3Δ χάρτη σε ένα ξεχωριστό PCD αρχείο για ανάλυση και επεξεργασία σε ύστερο χρόνο. Δημιουργήθηκε μια συνάρτηση που αποθηκεύει όλα τα τελικά σημεία του χάρτη και ενσωματώθηκε στην κλάση “system” του ORB-SLAM 2, η οποία διαχειρίζεται όλες τις βασικές λειτουργίες. Η συνάρτηση αυτή χρησιμοποιεί από την κλάση “MapPoint”, τη μεταβλητή “GetAllMapPoints” στην οποία είναι αποθηκευμένα όλα τα σημεία που δομούν το χάρτη. Το PCD αρχείο είναι ένας τύπος αρχείου που υποστηρίζει δεδομένα από 3Δ νέφη σημείων και ανήκει στη βιβλιοθήκη της PCL. Περιέχει κεφαλίδες που δηλώνουν ορισμένες ιδιότητες του νέφους σημείων όπως είναι η έκδοση της βιβλιοθήκης, το όνομα της κάθε διάστασης (χ, ψ, ζ συντεταγμένες), το μέγεθος (π.χ. 4 για unsigned float) και το είδος (π.χ. F για float) της κάθε διάστασης σε bytes, το συνολικό αριθμό των σημείων, κ.ά.
Το αρχείο των θέσεων της τροχιάς της κάμερας περιλαμβάνει το χρόνο καταγραφής της εικόνας-κλειδί, τις X, Y, Z συντεταγμένες και τα qx, qy, qz, qw που αποτελούν τον προσανατολισμό της ως προς ένα τετραδόνιο. Η σχέση που συνδέει το τετραδόνιο με το συμβατικό πίνακα στροφής είναι:

2*(qw**2+qx**2)-1	2*(qx*qy-qw*qz)	2*(qx*qz+qw*qy)
2*(qx*qy+qw*qz)	2*(qw**2+qy**2)-1	2*(qy*qz-qw*qx)
2*(qx*qz-qw*qy)	2*(qy*qz+qw*qx)	2*(qw**2+qz**2)-1

4.1.2 Δημιουργία αυξητικού οπτικού χάρτη

Σκοπός του οπτικού χάρτη είναι η οπτικοποίηση περιοχών ιδιαίτερα μεγάλων, έτσι ώστε να προσφέρει μια ενημερωτική εικόνα του περιβάλλοντος στο χρήστη και να επιτρέψει μια περαιτέρω μελέτη. Τα βασικά στάδια της προτεινόμενης μεθόδου είναι τα εξής:

- Διαβάζει τις εικόνες από το αρχείο που παράγει ο ORB-SLAM 2 μετά την ολοκλήρωση της εκτέλεσής του και αποθηκεύει τα ονόματά τους σε μια λίστα.
- Δημιουργεί ένα αρχείο με διαδοχικές εικόνες-«κλειδιά» και τις συνταυτίσεις τους (ανά δύο) χρησιμοποιώντας μια συνάρτηση του SLAM αλγορίθμου, την SeachForTriangulation. Η συνάρτηση αυτή υπολογίζει συνταυτίσεις μεταξύ ORB χαρακτηριστικών σημείων που δεν έχουν συνταυτιστεί της τρέχουσας εικόνας-«κλειδί» και ORB χαρακτηριστικών σημείων της προηγούμενης.
- Με βάση το αρχείο πορείας της κάμερας, εντοπίζονται στο προηγούμενο αρχείο συνταυτίσεις μεταξύ δύο διαδοχικών εικόνων. Ζεύγη εικόνων για τα οποία δεν υπάρχουν συνταυτίσεις ή υπάρχουν αλλά είναι λιγότερες από 4, χρησιμοποιείται ο ORB ανιχνευτής για την εξαγωγή τους.
- Υπολογίζονται οι διαδοχικές ομογραφίες μεταξύ των διαδοχικών εικόνων.
- Υπολογίζονται οι ομογραφίες με βάση την εικόνα αναφοράς και τα όρια των εικόνων μετασχηματίζονται με βάση αυτές.
- Εντοπίζεται η μικρότερη και μεγαλύτερη τιμή του χ και ψ και υπολογίζονται οι διαστάσεις της τελικής εικόνας ως η διαφορά των προηγούμενων.

- Οι ομογραφίες πολλαπλασιάζονται με μια μετατόπιση έτσι ώστε να μην “κόβεται” κάποια στην τελική εικόνα, λόγω του ότι η ομογραφία προβάλλει σημεία και στις αρνητικές τιμές της εικόνας.
- Προβάλλεται ολόκληρη η εικόνα στην εικόνα αναφοράς στις διπλάσιες διαστάσεις της τελική εικόνας.
- Η εικόνα βάφεται ως εξής: α) Διαβάζονται ένα ένα τα pixels της τελικής εικόνας. (β) Γίνεται η υπόθεση ότι είναι μαύρα. (γ) για κάθε pixel της ψάχνει στην αντίστοιχη θέση στις μετασχηματισμένες εικόνες όπου έχει χρώμα. Λόγω του ότι ψάχνει ένα ένα τα pixels της εικόνας, η διαδικασία είναι αρκετά κοστοβόρα. Έτσι, προτείνεται μια βελτίωση της μεθόδους ως εξής:
  Σε κάθε μετασχηματισμένη εικόνα εφαρμόζεται μια μάσκα η οποία ανιχνεύει σε ποιες περιοχές υπάρχει ή δεν υπάρχει χρώμα. Χρησιμοποιεί εκείνα τα pixels που έχουν χρώμα, για να βάψει τις αντίστοιχες θέσεις της τελικής εικόνας.

  Προκειμένου η διαδικασία να είναι αυξητική και πιο γρήγορη, σε κάθε μετασχηματισμένη εικόνα εφαρμόζεται μια μάσκα που ανιχνεύει που υπάρχει και δεν υπάρχει χρώμα. Στη συνέχεια, δημιουργήθηκε μια συνάρτηση που υπολογίζει τα όρια της έγχρωμης εικόνας και την κάνει διάφανη. Έτσι, με βάση τα υπολογισμένα όρια η τελική εικόνα βάφεται από την αντίστοιχη μετασχηματισμένη. Η διαδικασία αυτή πραγματοποιείται για κάθε ζευγάρι εικόνων και το αποτέλεσμα παρουσιάζεται. Παρόλο που το αποτέλεσμα παρουσιάζεται σωστό (χωρίς μαύρες περιοχές μεταξύ των εικόνων), η εικόνα δεν μπορεί να αποθηκευτεί. Η πληροφορία της διαφάνειας δεν μπορεί να μεταδοθεί σωστά. Ωστόσο, από τις δύο διαδικασίες που περιγράφηκαν παραπάνω, η εικόνα μπορεί να σωθεί.

4.2. Εφαρμογή ORB-SLAM 2

Για την επιτυχή εφαρμογή του αλγορίθμου ήταν απαραίτητη η εγκατάσταση κάποιων προαπαιτούμενων βιβλιοθηκών, όπως το Pangolin, OpenCV, Eigen, DBoW2 καθώς και το g2o. Αρχικά, το σύστημα εφαρμόστηκε για δύο γνωστά δεδομένα, μια αλληλουχία από το Πανεπιστήμιο TUM στο Μόναχο που περιγράφει μια αίθουσα γραφείου και μια αλληλουχία σε ένα βιομηχανικό περιβάλλον με χρήση ενός μικρού εναέριου οχήματος (MAV), στο Πανεπιστήμιο ETH της Ζυρίχης, προκειμένου να ελεγχθεί αν λειτουργεί σωστά το σύστημα. Για να εφαρμοστεί αποτελεσματικά, πρέπει να παρέχονται ένα ORB λεξικό (vocabulary), ένα αρχείο ρυθμίσεων και ο χρόνος λήψης των εικόνων (timestamps). Το ORB λεξικό δημιουργείται, μετατρέποντας τους δυαδικούς περιγραφείς (binary descriptors) σε “οπτικές” λέξεις (visual words). Το αρχείο ρυθμίσεων είναι απαραίτητο για τη μετάδοση όλων των παραμέτρων στον αλγόριθμο. Περιλαμβάνει τις παραμέτρους του εσωτερικού προσανατολισμού όπως αυτές προκύπτουν από τη βαθμονόμηση, τον αριθμό των χαρακτηριστικών σημείων που θα εξαχθούν σε κάθε εικόνα, μερικές παραμέτρους που αφορούν την οπτικοποίηση του αποτελέσματος (π.χ. το μέγεθος της εικόνας-κλειδί, το μέγεθος της γραμμής που τη περιγράφει, κ.ά.), τον αριθμό καρέ ανά δευτερόλεπτο κ.ά. Πολλές από αυτές τροποποιούνται ανάλογα την εκάστοτε εφαρμογή, όπως ο αριθμός των σημείων όταν η αρχικοποίηση δεν μπορεί να πραγματοποιηθεί λόγω του μικρού αριθμού τους. Στην περίπτωση που οι εικόνες εισόδου έχουν ήδη διορθωθεί, οι παράμετροι της ακτινικής και εφαπτομενικής διαστροφής θέτονται μηδέν. Εφόσον, το ORB-SLAM 2 έτρεξε σωστά με τα δύο ευρέως γνωστά δεδομένα, επιλέχθηκαν δύο αλληλουχίες που περιγράφουν δύο διαφορετικά περιβάλλοντα. Η πρώτη απεικονίζει ένα ενάλιο περιβάλλον, ενώ η δεύτερη ένα τμήμα αστικού ιστού. Τα πειράματα αφορούν τη μονοεικονική εφαρμογή του συστήματος.

4.2.1. Εφαρμογή αλγορίθμου σε υποθαλάσσιο περιβάλλον – AQUALOC Dataset

Επισκόπηση των δεδομένων:
Το AQUALOC δημιουργήθηκε από τους Ferrera et al. (2018). Οι εικόνες σε ανάλυση 968×608 δείχνουν έναν αρχαιολογικό χώρο, ένα ανάχωμα αρχαίων αμφορέων, όπου η κορυφή του είναι λίγα μέτρα πάνω από τον πυθμένα της θάλασσας της Μεσογείου, έξω από την ακτή της Κορσικής. Βρίσκεται περίπου 380 μέτρα κάτω από την επιφάνεια της θάλασσας και επομένως, χαρακτηρίζεται από δυναμικότητα (ύπαρξη θαλάσσιας πανίδας) και ποικιλία επιφανειών, όπως περιοχές με έντονη υφή (αμφορείς) και περιοχές χωρίς υφή (αμμώδεις). Τα δεδομένα συλλέχθηκαν, χρησιμοποιώντας ένα ROV, πάνω στο οποίο είχε τοποθετηθεί μια μονοχρωματική κάμερα με ένα fisheye φακό, ένας αισθητήρας πίεσης, ένα χαμηλού κόστους IMU σύστημα και ένας ενσωματωμένος υπολογιστής για την ταυτόχρονη καταγραφή των μετρήσεων του αισθητήρα. Η κάμερα είχε τοποθετηθεί πίσω από ένα θόλο με σκοπό την ελαχιστοποίηση των φαινομένων διαστροφής που παράγονται λόγω του διαφορετικού δείκτη διάθλασης και νερού. Ο ρυθμός καταγραφής των δεδομένων ήταν 20 Hz. Ακόμα, παρέχονται πραγματικά δεδομένα που έχουν προέλθει από μια κλασική φωτογραμμετρική διαδικασία εκτός σύνδεσης, καθώς και ο χρόνος λήψης των δεδομένων. Η συνολική διάρκεια είναι 569 δευτερόλεπτα, ενώ το μήκος του, 122.1 μέτρα.

Περιγραφή του αρχαιολογικού χώρου:
Ο αρχαιολογικός χώρος είναι γνωστός ως Capo Sagro 3 και είναι ένας από τα λίγα αρχαία ναυάγια της βαθιάς θάλασσας που δεν έχουν υποστεί ζημιά. Ο λόφος των αμφορέων έχει μήκος 16 μέτρων και πλάτος 9 μέτρων. Οι αμφορείς κατασκευάστηκαν για τη μεταφορά κρασιού κατά μήκος της ακτής της Τυρρηνικής Ιταλίας γύρω στο 180-170 π.Χ. Ο χώρος αυτός χρησιμοποιήθηκε για φωτογραμμετρική έρευνα και πολλοί αμφορείς ανακτήθηκαν για ανάλυση κατά το 2015-2016.

Εφαρμογή του ORB-SLAM 2:
Αρχικά, δημιουργήθηκε ένα εκτελέσιμο αρχείο που περιείχε τη λειτουργία αποθήκευσης του 3Δ χάρτη και τροποποιήθηκε το αρχείο ρυθμίσεων με βάση τις παραμέτρους βαθμονόμησης και τα χαρακτηριστικά του συγκεκριμένου περιβάλλοντος. Λαμβάνοντας υπόψιν το περιεχόμενο των εικόνων, την ανάλυσή τους και τις δυνατότητες του ORB περιγραφέα, ο αριθμός των εξαγόμενων σημείων επιλέχθηκε ίσος με 2000, προκειμένου να πραγματοποιηθεί αποτελεσματικά η αρχικοποίηση (σε 10 δευτερόλεπτα). Στην αρχή, επιλέχθηκαν 500 σημεία για την εξαγωγή, αλλά το σύστημα δεν μπόρεσε να κάνει την αρχικοποίηση. Στη συνέχεια, με 1000 χαρακτηριστικά σημεία, η αρχικοποίηση επετεύχθη αλλά μετά από 68 δευτερόλεπτα (επομένως πληροφορία από τις πρώτες εικόνες χάθηκε). Ο ORB-SLAM 2 χρειάζεται βαθμονομημένες εικόνες αλλά όχι απαραίτητα και διορθωμένες από τη διαστροφή. Οι εικόνες δεν διορθώθηκαν καθώς η χρήση τους προσφέρει μεγάλο υπολογιστικό κόστος, δυσχεραίνοντας την εφαρμογή του αλγορίθμου σε πραγματικό χρόνο. Για το λόγο αυτό, ο ORB-SLAM 2 χρησιμοποιεί τις παραμέτρους της διαστροφής και διορθώνει μόνο τα χαρακτηριστικά σημεία. Τέλος, δημιουργήθηκε ένα αρχείο που περιλάμβανε τα ονόματα των εικόνων και το χρόνο καταγραφής τους, προκειμένου το σύστημα να τις επεξεργαστεί στη σωστή σειρά. Ο χρόνος εκτέλεσης του αλγορίθμου ήταν 18’ 34’’. Ο μέσος χρόνος παρακολούθησης (mean tracking time) ήταν 65 ms και ο διάμεσος, 70 ms. Γενικά, ο χρόνος παρακολούθησης περιλαμβάνει την εξαγωγή χαρακτηριστικών σημείων ORB, την αρχική εκτίμηση της θέσης της κάμερας και την παρακολούθηση του τοπικού χάρτη, που αποτελεί την πιο απαιτητική διαδικασία. Ο χρόνος μπορεί να ελαττωθεί με τη μείωση των εικόνων-κλειδιά που περιλαμβάνονται στον τοπικό χάρτη. Εξαρτάται από τον αριθμό των εξαγόμενων σημείων, την ανάλυση της εικόνας και τη διαδικασία του relocalization, εάν υπάρξει αποτυχία στον υπολογισμό της θέσης. Κατά την εφαρμογή του αλγορίθμου δεν υπήρξε relocalization αλλά ο αριθμός των σημείων και η ανάλυση των εικόνων ήταν πολύ υψηλή. Επίσης, παρατηρήθηκε ότι ο ORB-SLAM 2 έβρισκε αρκετές συνταυτίσεις σημείων, παρά τις διαταραχές, τις δυναμικές αλλαγές και τις περιοχές χωρίς υφή, όπως ο βυθός. Στην τελευταία περίπτωση ο αλγόριθμος έβρισκε σημεία ως αποτέλεσμα μικρών διαφορών έντασης. Παρ’ όλα αυτά, λόγω των

ιδιαίτερων χαρακτηριστικών του περιβάλλοντος που δυσχεραίνουν την ομαλή λειτουργία του συστήματος, πολλά αποτελούν θόρυβο. Ένα από αυτά είναι η οπισθοσκέδαση, που δημιουργούσε λευκές κηλίδες πάνω στο απεικονιζόμενο αντικείμενο και η οποία προκαλείται από τα στροβοσκόπια που είναι ενσωματωμένα στο ROV και είναι στραμμένα προς το αντικείμενο. Για να αποφευχθεί το φαινόμενο αυτό, πρέπει τα στροβοσκόπια να είναι τοποθετημένα έτσι, ώστε το παραγόμενο φως να δημιουργεί έναν "κώνο" και οι αμφορείς να φωτίζονται από τα άκρα του. Ακόμη, στις εικόνες παρατηρούνται αμμώδη σύννεφα, θολότητα και έντονη δυναμικότητα λόγω της κίνησης της θαλάσσιας ζωής μπροστά στο φακό. Κατά την πλοήγηση του ROV η κάμερα πραγματοποίησε αρκετές μεταφορές και περιστροφές, ενώ κάποιες φορές πλησίαζε το αντικείμενο και άλλες απομακρυνόταν. Σε καμία από αυτές τις περιπτώσεις, δεν έχασε τη θέση της. Κατά τη 3Δ ανακατασκευή του χάρτη, παρατηρήθηκε ότι δημιουργούνταν πολλά διπλά σημεία λόγω του σφάλματος της κλίμακας. Αυτό προέρχεται από τη χρήση μονοεικονικής κάμερας, που οι μετρήσεις βάθους είναι ανύπαρκτες. Όπως αναφέρθηκε στην περιγραφή του αλγορίθμου, ο ORB-SLAM 2 περιλαμβάνει δύο πολύ βασικές διαδικασίες, το κλείσιμο του βρόγχου και τη συνολική βελτιστοποίηση του χάρτη και της πορείας της κάμερας. Λόγω αυτών, τα διπλά σημεία συγχωνεύτηκαν και το συσσωρευμένο σφάλμα διορθώθηκε, επιτυγχάνοντας τη συνολική γεωμετρική συνοχή. Επίσης, παρατηρήθηκε ότι πολλά σημεία που περιέγραφαν ψάρια, διαγράφονταν λίγο μετά τη δημιουργία τους, ενισχύοντας την ανθεκτικότητα του αλγορίθμου στο θόρυβο. Σε γενικές γραμμές, το σύστημα παρείχε ικανοποιητικά αποτελέσματα, λαμβάνοντας υπόψη τη συνολική γεωμετρία του αντικειμένου και την πορεία του ROV. Η υπολογισμένη πορεία του αποτελείτο από 654 εικόνες-κλειδιά έναντι των 11377 συνολικών εικόνων που χρησιμοποιήθηκαν (5.75%). Όπως αναφέρθηκε προηγουμένως, ο αλγόριθμος απορρίπτει τις εικόνες που δεν προσθέτουν αρκετή καινούργια πληροφορία, μειώνοντας το υπολογιστικό κόστος της συνόρθωσης.

Προκειμένου να αξιολογηθεί η απόδοση του συστήματος και η ακρίβεια υπολογισμού της θέσης, η παραγόμενη τροχιά συγκρίθηκε με την πραγματική. Οι δύο τροχιές αναφέρονταν σε διαφορετικό σύστημα αναφοράς. Αυτή που παράγεται από το σύστημα SLAM αναφέρεται σε τυχαίο σύστημα, όπου η κλίμακα επιλέγεται να είναι το μέσο βάθος του αρχικού χάρτη. Γι' αυτό, αρχικά αναφέρθηκαν στο ίδιο σύστημα συντεταγμένων και μετά συγκρίθηκαν. Ο μετασχηματισμός που εφαρμόστηκε στην παραγόμενη πορεία του ROV είναι ένας μετασχηματισμός ομοιότητας που έχει 7 βαθμούς ελευθερίας, αφού το σφάλμα μπορεί να εμφανιστεί στη μεταφορά, τη στροφή και την κλίμακα. Για τη συσχέτιση, αξιοποιήθηκε η μέθοδος του Horn (Absolute Orientation Problem, 1987), που χρησιμοποιεί 3 σημεία, γνωστά και στα δύο συστήματα, για τον υπολογισμό των 7 παραμέτρων. Λαμβάνει υπόψιν του μόνο τη μεταφορά και όχι τον προσανατολισμό, καθώς το σύστημα SLAM είναι ένας διαδοχικός αλγόριθμος και έτσι, τα σφάλματα στροφής θα εμφανιστούν με τη μορφή σφαλμάτων μεταφοράς αργότερα. Για την αξιολόγηση, δύο μεγέθη υπολογίστηκαν, το απόλυτο σφάλμα μεταφοράς και το σφάλμα σχετικής θέσης. Το πρώτο μέγεθος είναι κατάλληλο για την ακρίβεια του αλγορίθμου, ενώ το δεύτερο για τη μέτρηση της τοπικής και μακροπρόθεσμης συνοχής.

Το απόλυτο σφάλμα μεταφοράς (Α.Σ.Μ) υπολογίζει τη διαφορά μεταξύ ομόλογων σημείων στην εκτιμώμενη και την πραγματική τροχιά. Παρέχει μια μόνο μέτρηση για την εκτίμηση της θέσης και του προσανατολισμού και είναι ευαίσθητο στο χρόνο, στον οποίο εμφανίζεται το σφάλμα. Για παράδειγμα, αν το σφάλμα εμφανίζεται στο τέλος της πορείας, τότε το Α.Σ.Μ είναι μικρότερο. Αντίθετα, αν το σφάλμα εμφανιστεί στην αρχή, το Α.Σ.Μ θα είναι μεγαλύτερο, καθώς η λάθος εκτίμηση της αρχικής θέσης θα έχει ως αποτέλεσμα να διαδοθεί και στις επόμενες. Σχεδιάζοντας τις δύο τροχιές, παρατηρήθηκε σε γενικές γραμμές ότι η παραγόμενη ακολουθούσε την πραγματική πορεία. Σε ορισμένα σημεία του γραφήματος το σφάλμα ήταν μεγαλύτερο, κυρίως σε στροφές. Επειδή ο αλγόριθμος είναι μη ντετερμινιστικός, η εφαρμογή πραγματοποιήθηκε 9 φορές προκειμένου να εξαχθεί η ενδιάμεση τιμή για κάθε στατιστικό δείκτη (μέσο τετραγωνικό σφάλμα, μέσο σφάλμα, τυπική απόκλιση, κ.ά.). Από τους δείκτες παρατηρήθηκε ότι η ακρίβεια υπολογισμού της θέσης ήταν λίγο διαφορετική για κάθε εκτέλεση. Αυτό οφείλεται στην τυχαία αρχικοποίηση του συστήματος, που εξηγεί και το γεγονός της διαφορετικής κλίμακας σε κάθε προσπάθεια. Η ενδιάμεση τιμή σφάλματος ήταν περίπου 80 εκ. Η τιμή αυτή αντικατοπτρίζει το σφάλμα που παράγεται με την πάροδο του χρόνου, το οποίο προκαλεί τη θέσης της κάμερας να αποκλίνει από την πραγματική της τιμή.

Το μονοεικονικό ORB-SLAM 2 βασίζεται στην οπτική οδομετρία για τον υπολογισμό της θέσης, που προέρχεται από τη διαφορά της τρέχουσας εικόνας από την προηγούμενη. Αυτό, οδηγεί σε μια σταδιακή αλλαγή της θέσης και άρα, σε συσσωρευμένο σφάλμα. Παρ' όλο που ο αλγόριθμος ανιχνεύει, κλείνει βρόγχους και πραγματοποιεί συνολικές βελτιστοποιήσεις, το σφάλμα δεν μπορεί να εξαλειφθεί, ιδίως όταν η κάμερα πραγματοποιεί στροφές και ανιχνεύονται μεγάλοι βρόγχοι από διαφορετική οπτική γωνία.
Το σφάλμα σχετικής θέσης (Σ.Σ.Θ) υπολογίζει τη σχετική κίνηση μεταξύ δύο θέσεων στην εκτιμώμενη και την πραγματική τροχιά και είναι λιγότερο ευαίσθητο στο χρόνο που εμφανίζεται το σφάλμα. Για τον υπολογισμό του Σ.Σ.Θ επιλέχθηκαν θέσεις που απείχαν μικρή απόσταση μεταξύ τους για να αξιολογηθεί η τοπική συνοχή καθώς και θέσεις με μεγαλύτερη απόσταση, για να εκτιμηθεί η συνολική συνεκτικότητα. Από τις μετρήσεις, παρατηρήθηκε ότι το ενδιάμεσο σφάλμα άλλαζε ελάχιστα. Αυτό δείχνει τη γενικότερη συνοχή που επιτυγχάνει το σύστημα. Η συνολική ακρίβεια και αντοχή του αλγορίθμου εξαρτάται από πολλούς παράγοντες που μπορούν να χωριστούν σε δύο βασικές κατηγορίες, το υποθαλάσσιο περιβάλλον και το σχεδιασμό του ORB-SLAM 2. Όσον αφορά την πρώτη, οι δυναμικές αλλαγές (π.χ. ψάρια), οι δύσκολες συνθήκες, όπως η θόλωση και η σκέδαση, καθώς και οι περιοχές χωρίς υφή δυσκολεύουν την ομαλή λειτουργία της εξαγωγής χαρακτηριστικών σημείων και τη συνταύτισή τους. Αυτό έχει ως αποτέλεσμα τη λανθασμένη εκτίμηση της θέσης ακόμα και αν πολλά από αυτά απορριφθούν μέσω της συνόρθωσης. Όσον αφορά τον ORB-SLAM 2, όπως αναφέρθηκε, στηρίζεται στην οδομετρία και επομένως, το σφάλμα ολοένα και μεγαλώνει με την αύξηση της απόστασης. Χάρη στη λειτουργία κλεισίματος βρόγχων και τον περιορισμό της συνολικής βελτιστοποίησης σε ένα γράφημα που δεν λαμβάνει υπόψη τις μετρήσεις του αισθητήρα και διανέμει το σφάλμα κατά μήκος του, η ακρίβεια υπολογισμού διατηρείται σχετικά υψηλή. Επίσης, πρέπει να σημειωθεί ότι η πραγματική τροχιά του ROV υπολογίστηκε από μια κλασική φωτογραμμετρική μεθοδολογία χωρίς την ύπαρξη σημείων γνωστών συντεταγμένων και γι' αυτό δεν μπορεί να θεωρηθεί τόσο ακριβής, παρά το σχετικά χαμηλό σφάλμα επαναπροβολής (0.645 px). Ακόμα, ο φακός που χρησιμοποιήθηκε για τη συλλογή των δεδομένων είναι fisheye. Ο ORB-SLAM 2 δεν μπορεί να διαχειριστεί τέτοιου είδους φακούς, παρά μόνο κάμερες οπής. Πιο συγκεκριμένα, κατά την αρχικοποίηση, υπολογίζει δύο γεωμετρικά μοντέλα, μία ομογραφία και ένα θεμελιώδη πίνακα, που είναι κατάλληλα μόνο για τις συμβατικές κάμερες. Δεν μπορεί να εκμεταλλευτεί την πληροφορία από όλη την εικόνα, καθώς η κάμερα οπής προβάλλει τα 3Δ σημεία σε ένα 2Δ επίπεδο.

4.2.2 Εφαρμογή αλγορίθμου σε αστικό περιβάλλον – Zürich Dataset

Επισκόπηση των δεδομένων:
Το δεύτερο σετ δεδομένων που χρησιμοποιήθηκε είναι ένα τμήμα από το Zürich Dataset που δημιουργήθηκε από τους Majdik et al. (2016). Οι βαθμονομημένες εικόνες που συλλέχθηκαν από μια κάμερα GoPro, απεικονίζουν προσόψεις κτηρίων στους δρόμους της Ζυρίχης. Η κάμερα είναι ενσωματωμένη σε ένα μικρό εναέριο όχημα (MAV) που πετούσε στο κέντρο των δρόμων σε χαμηλό ύψος 5-15 μ. Οι εικόνες είναι πολύ υψηλής ανάλυσης (1920×1080×3) και ο ρυθμός καταγραφής τους είναι 30 Hz. Τα δεδομένα παρέχουν μετρήσεις GPS, μετρήσεις από αδρανειακά συστήματα, εικόνες εδάφους Google-street-view, το χρόνο καταγραφής των εικόνων καθώς και τις πραγματικές μετρήσεις θέσης. Οι πραγματικές μετρήσεις προέκυψαν από μια κλασική φωτογραμμετρική διαδικασία χωρίζοντας τα δεδομένα σε 1 καρέ/δευτερόλεπτο και με τέτοιο τρόπο ώστε να περιλαμβάνουν βρόγχους. Ως αρχική θέση επιλέχθηκε η μέτρηση από το GPS. Παρ' όλο που οι δέκτες GPS παρέχουν ακριβή υπολογισμό θέσης, η ακρίβεια και η αξιοπιστία τους εξαρτάται από τον αριθμό των διαθέσιμων δορυφόρων και επομένως, οι μετρήσεις που προέρχονται από αυτούς δεν μπορούν να θεωρηθούν ως πραγματικές. Στα αστικά δίκτυα, το σήμα του GPS είτε υποβαθμίζεται από τα γειτονικά κτήρια (σφάλμα πολυανάκλασης) είτε διακόπτεται τελείως. Το αρχικό μήκος της αλληλουχίας είναι 2 χλμ.

Περιγραφή των δεδομένων:
Η αλληλουχία που επιλέχθηκε από τα συνολικά δεδομένα αποτελείται από δύο μη πολυσύχναστους δρόμους στο κέντρο της Ζυρίχης που χαρακτηρίζονται από ψηλά κτήρια και δέντρα.

Εφαρμογή του ORB-SLAM 2:
Λόγω του μεγάλου μήκους των αρχικών δεδομένων, δηλαδή του τεράστιου αριθμού εικόνων (περίπου 81000) και επομένως, της μεγάλης διάρκειας εφαρμογής του αλγορίθμου, επιλέχθηκε ένα μέρος αυτών. Η επιλογή έγινε με τέτοιο τρόπο, ώστε οι εικόνες να περιλαμβάνουν στροφές, επαναλαμβανόμενα μοτίβα στις προσόψεις των κτηρίων και δυναμικά αντικείμενα, όπως άνθρωποι. Συγκεκριμένα, επιλέχθηκαν 7714 εικόνες που κάλυπταν μια περιοχή μήκους 190 μ. Όπως και στην προηγούμενη περίπτωση, το αρχείο ρυθμίσεων τροποποιήθηκε με βάση τα στοιχεία βαθμονόμησης, την ανάλυση των εικόνων, τον φακό (fisheye) και το προς εξέταση περιβάλλον. Ακόμη, δημιουργήθηκε το αρχείο με τα ονόματα και το χρόνο λήψης των εικόνων. Η πρώτη δοκιμή έγινε, αλλάζοντας το μέγεθος των εικόνων έτσι ώστε να είναι το 30% των αρχικών. Ο αριθμός των καρέ/δευτερόλεπτο τέθηκε ίσος με 30 και ο αριθμός των εξαγόμενων σημείων ίσος με 2000. Παρ' όλο που η αρχικοποίηση πραγματοποιήθηκε σε μόλις 2 δευτερόλεπτα, το σύστημα έδωσε καταστροφικά αποτελέσματα τόσο για το χάρτη όσο και την πορεία της κάμερας. Στην αρχή της εφαρμογής, ο αλγόριθμος λειτουργούσε σωστά, αλλά με την αύξηση της απόστασης, το σφάλμα ολοένα και συσσωρευόταν. Κατά τη διάρκεια της εξερεύνησης, ο χάρτης είχε τοπική συνοχή και το αποτέλεσμα έμοιαζε με την πραγματικότητα. Ωστόσο, όσο ο χρόνος περνούσε, το σφάλμα μεγάλωνε, οδηγώντας σε ασυνέπεια. Ο παραγόμενος χάρτης καθώς και η πορεία της κάμερας ακολουθούσαν μια κυκλική κίνηση, που είχε ως αποτέλεσμα, μέρος της δημιουργούμενης σκηνής να "μπερδεύεται" με ένα άλλο τμήμα του χάρτη. Ένας από τους παράγοντες που προκάλεσαν το συγκεκριμένο φαινόμενο, είναι η υποβάθμιση της ποιότητας της εικόνας (μείωση του μεγέθους της) που οδηγεί σε μείωση της αποτελεσματικότητας της διαδικασίας εξαγωγής σημείων και της συνταύτισής τους. Εφόσον η λύση ήταν μακριά από την πραγματικότητα, δεν έγιναν περαιτέρω υπολογισμοί και αξιολογήσεις.
Ακόμα, ο αλγόριθμος δοκιμάστηκε να εφαρμοστεί με 20 καρέ/δευτερόλεπτο και εξαγωγή 3000 σημείων. Παρ' όλο που ο χρόνος εκτέλεσης μειώθηκε σημαντικά, το αποτέλεσμα ήταν ακόμα μπερδεμένο. Μετά από κάποιες δοκιμές, το μέγεθος των εικόνων εισαγωγής άλλαξε με τέτοιο τρόπο ώστε να περιλαμβάνει το 90% της πληροφορίας των αρχικών. Επιλέχθηκαν 30 καρέ/δευτερόλεπτο και εξαγωγή 2000 σημείων, όπως προτείνεται και από τους Mur-Artal et al. (2015). Οι τιμές των υπόλοιπων παραμέτρων παρέμειναν όπως είχαν οριστεί από τους δημιουργούς του αλγορίθμου. Η βαθμονόμηση των καμερών έγινε με βάση το μοντέλο της κάμερας οπής. Η αρχικοποίηση πραγματοποιήθηκε σε δύο δευτερόλεπτα, γεγονός που οφείλεται στην σημαντική μετάθεση της κάμερας και την απεικονιζόμενη σκηνή. Τα δύο γεωμετρικά μοντέλα που υπολογίζονται κατά την αρχικοποίηση χρειάζονται μεγάλη παράλλαξη προκειμένου να τριγωνοποιήσουν τα πρώτα σημεία. Επιπλέον, το περιβάλλον χαρακτηριζόταν από έντονη υφή και άρα μεγάλες διακυμάνσεις της έντασης του χρώματος. Ωστόσο, ο χρόνος εκτέλεσης ήταν μεγάλος, περίπου 38', ενώ ο μέσος χρόνος παρακολούθησης ήταν 170 ms. Ο μεγάλος χρόνος οφείλεται στην υψηλή ανάλυση των εικόνων καθώς και στην ισχύ του υπολογιστή. Κατά την εφαρμογή, ο αλγόριθμος επεξεργάστηκε επιτυχώς όλες τις εικόνες, χωρίς να χάσει τη θέση κάποιας, παρ' όλο που πραγματοποίησε στροφές. Αυτό προήλθε από τη χαμηλή ταχύτητα και τις "μαλακές" κινήσεις του MAV κατά τη διάρκεια της πλοήγησής του. Παρατηρώντας τις εικόνες, ήταν φανερό πως ήταν πλήρως παραμορφωμένες, καθώς συλλέχθηκαν με fisheye φακό και δεν υπέστησαν καμία περαιτέρω διόρθωση. Ο ORB-SLAM 2 έβρισκε πολλές συνταυτίσεις μεταξύ διαδοχικών εικόνων, με τις περισσότερες να είναι σωστές. Στην περίπτωση των επαναλαμβανόμενων μοτίβων, ο αλγόριθμος απέφευγε την συσχέτιση σημείων, καθώς υπήρχαν πολλοί πιθανοί περιγραφείς για ένα σημείο. Στην περίπτωση ανθρώπων που βρίσκονταν σε κίνηση, το σύστημα εξήγαγε χαρακτηριστικά σημεία και τα απέδιδε στο 3Δ χάρτη. Ωστόσο, λίγο μετά την τριγωνοποίησή τους, τα περισσότερα διαγράφονταν. Αντιθέτως, για ανθρώπους που παρέμειναν ακίνητοι, ο αλγόριθμος δημιουργούσε σημεία, τα οποία αποτελούσαν θόρυβο. Ακόμη, σημεία του χάρτη περιέγραφαν

δέντρα, σταθμευμένα αυτοκίνητα και άλλον αστικό εξοπλισμό, όπως κάδους απορριμμάτων και πινακίδες σήμανσης. Τα σημεία που περιέγραφαν κινούμενα αντικείμενα, όπως φύλλα δέντρων και πουλιά αποτελούν θόρυβο και δεν μπορούν να χρησιμοποιηθούν για τον ακριβή υπολογισμό της θέσης του MAV. Σε γενικές γραμμές, τα σημεία του 3Δ χάρτη περιέγραφαν καλά τις επιφάνειες και τα επίπεδα των κτηρίων, και το σχετικό βάθος των αντικειμένων αποδιδόταν σωστά. Επίσης, υπήρχαν διπλά σημεία τα οποία καθιστούσαν δύσληπτο το τελικό αποτέλεσμα. Λόγω της έλλειψης βρόγχων, τα σημεία δεν μπορούσαν να συγχωνευτούν και άρα ούτε να διορθωθεί το σφάλμα μεταφοράς, στροφής και κλίμακας. Οι συνθήκες για να ανιχνευτεί και να κλείσει ένας βρόγχος αλλά και για να ξεκινήσει η διαδικασία της συνόρθωσης δέσμης δεν πληρούνταν και ως εκ τούτου, τα 3Δ σημεία του χώρου και οι θέσεις της κάμερας δεν βελτιστοποιήθηκαν. Παρόμοια με την πρώτη εκτέλεση, ο τοπικός χάρτης ήταν συνεπής αλλά το σύστημα δεν μπορούσε να διατηρήσει τη συνολική συνοχή του. Ο δρόμος παρουσιαζόταν ως καμπύλη γραμμή αντί για ευθεία. Παρ' ολα αυτά, τα τελικά αποτελέσματα ήταν καλύτερα από την προηγούμενη προσπάθεια. Η παραγόμενη τροχιά αποτελείτο από 372 εικόνες-κλειδιά έναντι των 7714 εικόνων που χρησιμοποιήθηκαν συνολικά (4.82%). Όπως αναφέρθηκε, ο ORB-SLAM 2 απορρίπτει εικόνες οι οποίες προσφέρουν ελάχιστη καινούργια πληροφορία στο σύστημα. Αυτό παρατηρήθηκε και κατά τη διάρκεια της εφαρμογής, όπου ο αλγόριθμος δημιουργούσε εικόνες-κλειδιά και λίγο μετά τις διέγραφε. Η δυνατότητα αυτή βελτιώνει την ακρίβεια, αφού η διαδικασία χαρτογράφησης δεν είναι συνδεδεμένη με τον υψηλό ρυθμό καταγραφής καρέ (30). Κάθε 30 εικόνες, το οπτικό περιεχόμενο αλλάζει ελάχιστα. Ο αλγόριθμος κρατάει περίπου μία εικόνα-κλειδί ανά 30 εικόνες. Επίσης, μειώνει το υπολογιστικό κόστος της βελτιστοποίησης. Όταν η εκτιμώμενη πορεία σχεδιάστηκε, ήταν διαφορετική από αυτή που δημιουργείτο κατά την εξερεύνηση και σε διαφορετική κλίμακα από τον αντίστοιχο 3Δ χάρτη. Έτσι, δεν μπόρεσε να αξιολογηθεί η ακρίβεια συγκρίνοντας το αποτέλεσμα με την πραγματική τροχιά. Εξάλλου, το σφάλμα ήταν πολύ μεγάλο λόγω της έλλειψης βρόγχων.

4.2.3 Εφαρμογή οπτικού χάρτη

Για την αξιολόγηση των μεθόδων εκτελέστηκαν διάφορα πειράματα με χρήση διαφορετικού αριθμού εικόνων. Αρχικά, η μέθοδος εφαρμόστηκε για 4 εικόνες. Με βάση τις εικόνες που παρήχθησαν, η ευθυγράμμισή τους ήταν σωστή. Ωστόσο, όταν χρησιμοποιήθηκαν εικόνες όπου η κάμερα πραγματοποιούσε μετάθεση και στροφή, η μέθοδος δεν έδινε καλά αποτελέσματα. Αυτό οφείλεται στο ότι η προβολή των σημείων της εικόνας στην εικόνα αναφοράς τείνει προς το άπειρο, λόγω του ότι τα δύο επίπεδα μεταξύ τους δημιουργούν γωνία που πλησιάζει τις 90 μοίρες. Στη συνέχεια, για την παραγωγή του οπτικού χάρτη χρησιμοποιήθηκαν 23 εικόνες. Παρατηρώντας τα αποτελέσματα, φάνηκε ότι καθώς ο αριθμός των εικόνων αυξανόταν, το σφάλμα στην ευθυγράμμιση ήταν ολοένα και εντονότερο. Αυτό συνέβη, λόγω του συνεχόμενου πολλαπλασιασμού των ομογραφιών, με αποτέλεσμα τη συσσώρευση του σφάλματος. Έτσι, οι πρώτες 10-15 εικόνες ευθυγραμμίζονταν σωστά ενώ οι επόμενες εμφάνιζαν ασυνέχειες. Το ίδιο παρατηρήθηκε και στην εφαρμογή της μεθόδου με 70 εικόνες. Προκειμένου να βεβαιωθεί ότι οι συνταυτίσεις που προκύπτουν από τον ORB-SLAM 2 και τον ORB ανιχνευτή ήταν σωστές, σχεδιάστηκαν πάνω σε ζεύγη εικόνων. Για τον ποσοτικό έλεγχο αυτών υπολογίστηκε το μέσο τετραγωνικό τους σφάλμα ανάμεσα στις δύο διαδοχικές εικόνες. Σε όλα τα πειράματα, το σφάλμα αυτό ήταν περίπου 2-3 pixels. Επίσης, όσον αφορά το πείραμα των 70 εικόνων, όπου μετά από ένα σημείο χάνεται η ευθυγράμμιση, οι «λανθασμένες» εικόνες χρησιμοποιήθηκαν για την παραγωγή του οπτικού χάρτη από την αρχή. Τα αποτελέσματα έδειξαν ότι οι ευθυγραμμίσεις μεταξύ των εικόνων ήταν σωστές.

5. Συμπεράσματα/Μελλοντικές επεκτάσεις

Η παρούσα εργασία εστιάζει στην ανάπτυξη και εφαρμογή ενός μονοεικονικού συστήματος οπτικού SLAM που εφαρμόζεται σε πραγματικό χρόνο. Αρχικά, πραγματοποιήθηκε μια εκτεταμένη ανασκόπηση της βιβλιογραφίας, προκειμένου να βρεθεί και να αξιολογηθεί ένας κατάλληλος αλγόριθμος SLAM. Ο ORB-SLAM 2 που αναπτύχθηκε από τους Mur-Artal et al.

(2017) επιλέχθηκε ως ο καλύτερος αλγόριθμος που διατίθεται σήμερα. Στηρίζεται σε χαρακτηριστικά σημεία και παράγει ένα 3Δ χάρτη με βάση τη θέση ορισμένων εικόνων-κλειδιά. Ακόμη, υπολογίζει και αποθηκεύει την πορεία της κάμερας. Αποτελείται από τρεις βασικές διαδικασίες, την παρακολούθηση, τη χαρτογράφηση και το κλείσιμο βρόγχων που λειτουργούν ταυτόχρονα, χρησιμοποιώντας τα ίδια σημεία, με αποτέλεσμα τη σημαντική μείωση του υπολογιστικού κόστους και μια αποτελεσματική και αξιόπιστη διαδικασία. Κάνει χρήση του ORB περιγραφέα, ο οποίος είναι μια τάξη μεγέθους γρηγορότερος από τον SURF και δύο τάξεις, από τον SIFT. Η συγκεκριμένη διπλωματική παρέχει μια εις βάθος εξήγηση των βημάτων του αλγορίθμου.
Ταυτόχρονα με την πορεία της κάμερας, παράγει ένα αραιό και άχρωμο νέφος σημείων, το οποίο διαγράφεται λίγο μετά την ολοκλήρωση της εφαρμογής. Για αυτό το λόγο, δημιουργήθηκε μια συνάρτηση που αποθηκεύει το χάρτη σε ένα PCD αρχείο, με σκοπό την οπτικοποίηση και περαιτέρω αξιολόγησή του. Το συγκεκριμένο προϊόν, σε συνδυασμό με τις παραμέτρους του εξωτερικού προσανατολισμού (θέση κάμερας) μπορεί να συμβάλει στη δημιουργία ενός ακριβούς πυκνού νέφους σημείων, που μπορεί να χρησιμοποιηθεί σε πληθώρα εφαρμογών όπως, στην αναγνώριση εμποδίων, στην πλοήγηση σε έναν προκατασκευασμένο χάρτη κ.ά. Έτσι, αναπτύχθηκε ένας 2Δ αυξητικός οπτικός χάρτης με σκοπό να προσφέρει μια πιο ολοκληρωμένη εικόνα του εξεταζόμενου περιβάλλοντος στο χρήστη και ως εκ τούτου, την περαιτέρω ανάλυσή του.

5.1. ORB-SLAM 2

Για την εφαρμογή και αξιολόγηση του συστήματος με βάση την ακρίβεια υπολογισμού της θέσης, χρησιμοποιήθηκαν δύο σύνολα δεδομένων, σε διαφορετικά περιβάλλοντα. Το πρώτο απεικονίζει έναν υποβρύχιο αρχαιολογικό χώρο με αμφορείς σε βάθος 380 μέτρων, ενώ το δεύτερο καταγράφει δύο αστικούς δρόμους, στο κέντρο της Ζυρίχης, στην Ελβετία. Από το πρώτο πείραμα, βρέθηκε ότι η εξαγωγή 2000 χαρακτηριστικών σημείων οδήγησε σε μια γρήγορη και επιτυχή αρχικοποίηση του συστήματος (10 sec). Ο μέσος χρόνος παρακολούθησης ήταν 65 ms ενώ ο χρόνος εκτέλεσης, 18’ 34. Ο χρόνος μπορεί να ελαττωθεί με τη μείωση των εικόνων-κλειδιά που περιλαμβάνονται στον τοπικό χάρτη, αφού η παρακολούθηση του συγκεκριμένου χάρτη αποτελεί την πιο επίπονη διαδικασία του αλγορίθμου, όσον αφορά το υπολογιστικό κόστος. Κατά την εξερεύνηση, το σύστημα δεν “έχασε” τη θέση της κάμερας, παρά την οπτική υποβάθμιση και τα ιδιαίτερα χαρακτηριστικά του περιβάλλοντος. Αυτό προκλήθηκε από τον υψηλό ρυθμό καταγραφής (μεγάλος αριθμός εικόνων/δευτερόλεπτο) που έχει ως αποτέλεσμα την μικρή κίνηση μεταξύ των εικόνων επιτρέποντας έτσι μια ανθεκτική παρακολούθηση. Επίσης, ο ORB-SLAM 2 έβρισκε μεγάλο αριθμό ομόλογων σημείων μεταξύ των διαδοχικών εικόνων, με κάποια από αυτά να είναι λανθασμένα. Πιο συγκεκριμένα, σημεία που περιέγραφαν δυναμικές αλλαγές, όπως ψάρια, άλλαζαν γρήγορα θέση και ως εκ τούτου, η διατήρησή τους στη διαδικασία της συνταύτισης οδηγούσε σε λανθασμένη αντιστοιχία. Η λανθασμένη συνταύτιση υποβαθμίζει την ακρίβεια υπολογισμού της θέσης. Για αυτό το λόγο, ο αλγόριθμος μπορεί να συνδυαστεί με τεχνικές βαθιάς μάθησης και συγκεκριμένα, δίκτυα κατάτμησης που φιλτράρουν τα δυναμικά αντικείμενα και τα απορρίπτουν. Μια παρόμοια τεχνική έχει αναπτυχθεί από τους Yu et al. (2018), το DS-SLAM. Ακόμη, παρατηρήθηκε ότι υπήρχαν πολλά διπλά σημεία στην περιγραφή του χάρτη, οδηγώντας σε ένα λανθασμένο αποτέλεσμα. Το σύστημα δεν μπορεί να αντιληφθεί ότι έχει ξαναπεράσει από την ίδια περιοχή και ανακατασκευάζει από την αρχή το χάρτη. Αυτό, σε συνδυασμό με το συσσωρευμένο σφάλμα που παράγεται, οδηγεί στη δημιουργία διπλών σημείων. Παρόλα αυτά, ο αλγόριθμος διόρθωσε την ασυνέπεια, ενώνοντας τα δύο μέρη του βρόγχου, χάριν σε μια λειτουργία κλεισίματος βρόγχων. Η διαδικασία αυτή μπορεί να συνδυαστεί με συνελικτικά νευρωνικά δίκτυα (CNNs), με σκοπό να είναι αμετάβλητη στις περιβαλλοντικές αλλαγές και στα κινούμενα αντικείμενα.
Για την αξιολόγηση της πορείας της κάμερας, το παραγόμενο προϊόν συγκρίθηκε με το πραγματικό. Δεδομένου ότι ο ORB-SLAM 2 χρησιμοποιεί αυθαίρετο σύστημα συντεταγμένων, οι δύο τροχιές αναφέρθηκαν σε κοινό σύστημα χρησιμοποιώντας τη μέθοδο

του Horn. Ακόμη, λόγω του μη ντετερμινιστικού χαρακτήρα του, πραγματοποιήθηκαν 9 εκτελέσεις και εξήχθη η ενδιάμεση τιμή του σφάλματος. Η τιμή αυτή προέκυψε 0.078 μ., που είναι ικανοποιητική, λαμβάνοντας υπόψη το εξεταζόμενο περιβάλλον. Επιπλέον, πρέπει να αναφερθεί ότι ο ORB-SLAM 2 δεν είναι ικανός να διαχειριστεί εικόνες από φακό fisheye, καθώς παρέχει μοντέλο μόνο για κάμερες σημειακής οπής και επομένως, δεν μπορεί να εκμεταλλευτεί ολόκληρη την πληροφορία από τις εικόνες. Έτσι, προτείνεται η εφαρμογή του ORB-SLAM 3, που αποτελεί εξέλιξη του εξεταζόμενου αλγορίθμου και διαχειρίζεται τέτοιου τύπου εικόνες. Τα δεδομένα που χρησιμοποιήθηκαν παρέχουν μετρήσεις IMU, οι οποίες δεν μπόρεσαν να αξιοποιηθούν κατά τη διάρκεια της παρούσας διπλωματικής εργασίας. Ο ORB-SLAM 3 επιτρέπει την ενσωμάτωση αυτών των μετρήσεων στο οπτικό σύστημα SLAM, ενισχύοντας την αποτελεσματικότητα και την ακρίβειά του.

Από το δεύτερο πείραμα εξήχθη το συμπέρασμα ότι ο ORB-SLAM 2 είναι ευαίσθητος στις εικόνες υψηλής ανάλυσης, στο φακό fisheye και στην έλλειψη βρόγχων. Διατηρώντας την αρχική ανάλυση των εικόνων (1920×1080) ο αλγόριθμος επεξεργαζόταν τις εικόνες με πολύ αργό ρυθμό, αποκλείοντας την εφαρμογή σε πραγματικό χρόνο. Μετά από μερικές δοκιμές, παρατηρήθηκε ότι το σύστημα δεν μπορεί να διαχειριστεί εικόνες πολύ μικρότερης ανάλυσης, λόγω του ότι χάνεται αρκετή πληροφορία. Το παραγόμενο αποτέλεσμα τόσο του χάρτη όσο και της πορείας της κάμερας ήταν καταστροφικά. Γι' αυτό ευθύνεται και η μεγάλη διαστροφή που ήταν έντονη στις εικόνες και η οποία δεν είχε απαλειφθεί/περιοριστεί λόγω του μεγάλου υπολογιστικού κόστους. Ακόμη, πρέπει να σημειωθεί ότι η βαθμονόμηση της μηχανής έγινε με βάση το μοντέλο της κάμερας οπής. Θα μπορούσε να δοκιμαστεί χρησιμοποιώντας το μοντέλο του fisheye φακού, αλλά και πάλι δεν θα υπήρχε σημαντική βελτίωση στο αποτέλεσμα, αφού ο ORB-SLAM 2 διαχειρίζεται μόνο συμβατικές μηχανές. Το παραγόμενο αποτέλεσμα βελτιώθηκε όταν η ανάλυση της εικόνας έγινε τέτοια, έτσι ώστε να περιλαμβάνεται το 90% της αρχικής. Ενώ ο 3Δ χάρτης καθώς και η πορεία της κάμερας ακολουθούσαν σε γενικές γραμμές τη γεωμετρία του περιβάλλοντος και την κίνηση του MAV, οι δρόμοι απεικονίζονταν με μια μικρή καμπυλότητα. Ωστόσο, οι επιφάνειες, τα επίπεδα και το σχετικό βάθος μεταξύ των αντικειμένων αναπαρίστανται σωστά. Όπως αναφέρθηκε προηγουμένως, ο αλγόριθμος είναι ευαίσθητος στις δυναμικές αλλαγές. Για την αποφυγή των λανθασμένων συνταυτίσεων μεταξύ σημείων που κινούνται και επομένως, τη βελτίωση της ακρίβειας, ο παραγόμενος χάρτης μπορεί να ενισχυθεί με σημασιολογική πληροφορία. Ο αλγόριθμος μπορεί να στηριχτεί σε αυτή και να απορρίψει σημεία που περιγράφουν κινητά αντικείμενα. Όταν η υπολογισμένη πορεία της κάμερας σχεδιάστηκε, παρατηρήθηκε ότι το αποτέλεσμα ήταν διαφορετικό από εκείνο που δημιουργείτο κατά τη διάρκεια της εφαρμογής και είχε διαφορετική κλίμακα από τον 3Δ χάρτη. Ο λόγος για τον οποίο συμβαίνει αυτό χρειάζεται περαιτέρω διερεύνηση. Επομένως, το αποτέλεσμα δεν μπορούσε να αξιολογηθεί ποσοτικά και ποιοτικά.

Ακόμη, ο αλγόριθμος δοκιμάστηκε να εφαρμοστεί με περισσότερες εικόνες, προκειμένου να μελετηθεί η συμπεριφορά του σφάλματος. Πράγματι, όσο η απόσταση αυξανόταν, το σφάλμα γινόταν ολοένα και μεγαλύτερο, οδηγώντας σε απαγορευτικά αποτελέσματα. Αυτό αντικατοπτρίζει τη μεγάλη σημασία της ύπαρξης βρόγχων στο μονοεικονικό σύστημα SLAM, ιδίως σε μεγάλης διάρκειας εξερευνήσεις. Μια λύση που θα είχε μεγάλο ενδιαφέρον είναι η δημιουργία μιας συνάρτησης που θα φορτώνει τον παραγόμενο χάρτη στο σύστημα και θα λειτουργεί σε συνδυασμό με τη συνάρτηση αποθήκευσης. Αξιοποιώντας τη λειτουργία εντοπισμού του ORB-SLAM 2, το σύστημα θα μπορεί να φορτώσει τον ήδη κατασκευασμένο χάρτη και να εντοπίσει τη θέση της τρέχουσας εικόνας μέσα σε αυτόν. Έτσι, η συνεχής ακρίβεια εντοπισμού μπορεί να βελτιωθεί, επιτρέποντας μια μεγάλης κλίμακας ανακατασκευή. Σύμφωνα με τους Mur-Artal et al. (2017), η λειτουργία αυτή προϋποθέτει ότι η περιοχή έχει χαρτογραφηθεί καλά και ότι το υπό εξέταση περιβάλλον δεν παρουσιάζει σημαντικές αλλαγές. Κατά τη διάρκεια της διαδικασίας αυτής, οι διαδικασίες της χαρτογράφησης και του κλεισίματος βρόγχων είναι απενεργοποιημένες.

Επιπλέον, έγινε μια δοκιμή με διαφορετική αλληλουχία εικόνων, που περιλάμβανε κάποιες στροφές της κάμερας, Ο αλγόριθμος επεξεργαζόταν σωστά τις εικόνες, μέχρι ένα σημείο που έχασε την παρακολούθηση. Επειδή, η συγκεκριμένη αλληλουχία δεν περιλάμβανε κάποιο βρόγχο, το σύστημα προσπαθούσε να υπολογίσει τη θέση της τρέχουσας κάμερας χωρίς επιτυχία. Αυτό είχε ως αποτέλεσμα, να αγνοήσει όλες τις υπόλοιπες εικόνες. Σε τέτοιες

περιπτώσεις, μια αποτελεσματική λύση θα ήταν ο αλγόριθμος να τερματίζει τη διαδικασία μετά από κάποιο αριθμό εικόνων που δεν μπόρεσε να υπολογίσει τη θέση τους και να ξεκινά πάλι από την αρχή. Μπορεί να κρατάει στη μνήμη του "υπο-χάρτες" προκειμένου να μη χάσει όλες τις προηγούμενες εκτιμήσεις. Πιο συγκεκριμένα, μπορούν να τρέχουν παράλληλα δύο διαδικασίες. Η πρώτη θα προσπαθεί να υπολογίσει τη θέση της τρέχουσας εικόνας σε περίπτωση που υπάρξει κάποια περιοχή από την οποία έχει ξαναπεράσει η κάμερα, και η δεύτερη θα πραγματοποιεί την αρχικοποίηση. Η διαδικασία του κλεισίματος βρόγχων είναι σημαντική για τη διόρθωση του σφάλματος που θα προκύψει ανάμεσα στους "υπο-χάρτες". Οι "υπο-χάρτες" θα έχουν διαφορετική κλίμακα, αφού ο αλγόριθμος είναι μη-ντετερμινιστικός και η αρχικοποίηση γίνεται τυχαία. Γι' αυτό, μπορεί να εφαρμοστεί ο ORB-SLAM 3 που περιλαμβάνει αυτή τη λειτουργία.
Το κύριο πρόβλημα που παρατηρήθηκε είναι το σφάλμα που συσσωρεύεται με την αύξηση της διανυόμενης απόστασης. Η κλίμακα είναι άγνωστη και επομένως, το προϊόν που παράγεται είναι σε τυχαία κλίμακα. Τόσο οι μετρήσεις GPS, όσο και άλλες όπως αυτές που προκύπτουν από αδρανειακά συστήματα, μπορούν να ενσωματωθούν στο σύστημα. Αναλυτικότερα, στη διαδικασία της παρακολούθησης, μπορούν να αντικαταστήσουν το μοντέλο πρόβλεψης της κίνησης, κάνοντας τη διαδικασία πιο γρήγορη και αποτελεσματική.
Μελλοντικά ο αλγόριθμος θα μπορούσε να ελεγχθεί κατά την εκτέλεσή του σε πραγματικό χρόνο.

5.2 Οπτικός χάρτης

Όσον αφορά τον οπτικό χάρτη, από τα πειράματα εξήχθη το συμπέρασμα ότι οι δύο προτεινόμενες μέθοδοι δίνουν σωστά αποτελέσματα για ένα συγκεκριμένο αριθμό εικόνων. Όταν ο αριθμός των εικόνων αυξάνεται, το σφάλμα στην εκτίμηση των ομογραφιών συσσωρεύεται. Αυτό οφείλεται στο γεγονός ότι ως εικόνα αναφοράς χρησιμοποιείται η πρώτη εικόνα. Τα σφάλματα ακόμα προκύπτουν από τη μη επιπεδότητα του αντικειμένου καθώς και των στροφών της κάμερας που πραγματοποιεί το ROV. Για τον περιορισμό των σφαλμάτων μπορεί να εξεταστεί η χρήση της μεσαίας εικόνας ως εικόνα αναφοράς. Επιπλέον, για τη μείωση του σφάλματος, οι υπόλοιπες συναρτήσεις που χρησιμοποιεί ο ORB-SLAM 2 μπορούν να μελετηθούν έτσι ώστε να βρεθεί αν υπάρχει τρόπος, ο οπτικός χάρτης να δημιουργείται από εικόνες και όχι αναγκαστικά εικόνες-«κλειδιά» με συγκεκριμένο βήμα. Ακόμη, παρατηρήθηκαν ραφές μεταξύ των ευθυγραμμισμένων εικόνων, οι οποίες μπορούν να ανιχνευτούν και να περιοριστούν, κάνοντας τις επικαλυπτόμενες περιοχές πιο ομαλές και δίνοντας ένα πιο κατανοητό αποτέλεσμα. Στην περίπτωση μεγάλου αριθμού εικόνων, μετά από ένα σημείο και ύστερα οι εικόνες «κόβονται» διότι οι διαστάσεις της τελικής εικόνας δεν επαρκούν. Για το λόγο αυτό, προτείνεται η δημιουργία ενός γραφικού περιβάλλοντος που θα λειτουργεί κατά την εκτέλεση του ORB-SLAM 2 και θα αλλάζει αυτόματα τις διαστάσεις ανάλογα το μέγεθος των εικόνων.

1. Introduction-Scope of study

With the rapid improvement of technology in the field of photogrammetry, computer vision and automation, more complex and challenging tasks can be controlled and addressed by people. In recent years, the rapid and intense rhythms of life have generated the need of a more automatic and fast performance of things. Photogrammetry is the process of extracting metric information from images, i.e., the location and shape of objects. The integration of photography in surveying, primarily facilitates the production of maps. Computer vision (CV) is a field that focuses on interpreting and understanding the content of digital images and videos automatically. Nowadays, computers can efficiently and accurately identify and classify objects. However, it is not a trivial task as it is for humans and hence, further research is needed.

The image-based techniques for measuring and 3D modeling have aroused great interest in recent years, since new tools which increase the automation of photogrammetric processes, have been developed. Cameras offer low power consumption, enriched visual information and simplicity for implementation, compared to other existing sensors. Structure from Motion (SfM) is the most common method for 3D reconstruction due to its efficiency and low cost. It determines the spatial and geometric relationship of the target object through the movement of the camera. It can simultaneously determine both the parameters of the camera and 3D structure of a scene by combining 2D images taken from different viewpoints (Karmacharya et al. 2019). It can also deal with different cameras and an unordered set of images. However, the image-based 3D reconstruction requires much time to acquire or process the image data due to their high computational complexity, i.e., O(n^4) with respect to the number of cameras (Wu, 2013), something that hampers the real-time performance. Recent advancements of Visual SLAM (VSLAM) make the 3D reconstruction of a map in real-time feasible, in varying environments, from small indoor to large outdoor scenarios, assuming that the calibration of the camera has already been performed. In our era, VSLAM has undergone a remarkable evolution, providing speed and scalability, which constitutes a disruptive technology for surveyors. Visual SLAM refers to a technique of defining the position and orientation of a sensor with respect to its surroundings, while 3D reconstructing or mapping its environment, simultaneously. The geometric relationship between the environment and the sensor can be determined from the ordered sequences of images. Therefore, Visual SLAM is increasingly becoming a major breakthrough in embedded vision, as camera-based motion estimation is gaining immense popularity due to its simplicity and the use of limited resources in producing motion trajectories. Visual SLAM is used for a plethora of applications in the field of robotics, autonomous driving, augmented reality and terrestrial and underwater cultural heritage that become more commercially viable (Association for Advancing Automation, 2018).

Specifically, with the rapid developments in mobile robotics and industrial automation, there has been an increasing need for precise navigation and localization of moving platforms (Singandhupe et al. 2019). Currently, robots are capable of performing demanding and perplexing tasks autonomously, while in the past a human intervention was necessary. Mobile robotics plays a significant role in many applications, like medicine, military, space (e.g., rovers and landers for exploring Mars use Visual SLAM systems to navigate), agriculture (e.g., drones as well as robots can independently move around crops), autonomous driving, etc. (Yousif et al. 2015). Hence, autonomous navigation and localization is required. The simplest and oldest solution in localization problem is the use of wheel odometry. The current location of a robot is estimated, using wheel rotation measurements in combination with robot's motion model. However, this method generates some limitations such as the accumulative errors which lead to drift. Thus, the enhancement of visual odometry and SLAM systems into mobile robotics is of great importance.

Augmented reality is a developing technology which is used to increase the gaming experience as well as the productivity of businesses and industries. Since SLAM systems can understand

the physical world through feature points, they can also contribute to the improvement of augmented reality (AR). This enables AR applications to identify objects in 3D space and monitor the environment instantly, enhancing it with digital interactive "augmentations" (Wikitude SLAM technology). Nevertheless, the scene recognition and the sensor's pose estimation constitute difficult tasks because of noise existence, the changes of illumination and complicated backgrounds in images. Therefore, there is a need of a carefully designed SLAM algorithm which can be combined with Machine Learning techniques (ML).

Cultural heritage is the cultural legacy of physical artifacts and intangible attributes of a group or society that are inherited from past generations, preserved in the present and bestowed for the benefit of future generations (Ramirez 2017, UNESCO). Cultural heritage is unique and irreplaceable and thus, the documentation, preservation, and protection of it, are of significant importance. It is divided into cultural and natural heritage. The former includes tangible (i.e., paintings, sculptures, coins, monuments, archaeological sites, shipwrecks, underwater ruins etc.) and intangible cultural heritage (i.e., oral traditions, performing arts, rituals etc.). The latter contains cultural landscapes, physical, biological, or geological formations, etc. From the foregoing, it can be deduced that precise documentation of cultural heritage is crucial for its protection, conservation (e.g., monitoring) and for studies conducted during the renovation and restoration processes. Digitization is essential, as it can preserve tangible heritage, support education, enhance museums' competitiveness by boosting visitors' experience and allow infinite reach for everyone. Digital photogrammetry contributes to understanding and documenting historical and architectural objects as well as providing their geometrical and morphological characteristics (Pulcrano, 2019).
Visual SLAM systems enhanced by different types of sensors allow the real-time documentation of objects and space around them, in detail and accuracy and can effectively compete the existing conventional photogrammetric processes which have heavy computational cost. Moreover, some Visual SLAM systems can reduce the human intervention by choosing images automatically from a continuous capture, which allows the user to think less about using the camera or taking a picture, or by integrating SLAM with automatic vehicles such as UAVs or MAVs and ROVs. With integrated SLAM, areas that are inaccessible to manned vehicles can be sensed. With on-board cameras, UAVs can take aerial images during flight and produce 3D maps through image-based post-processing techniques. Furthermore, SLAM especially with ROVs is very attractive in processes in underwater environments, since the former has very high-performance speed and GPS is unable to work in such scenarios. Recent applications have shown the great potentials of using such vehicles to monitor the marine life, reconstruct the deep underwater seabed, inspect sunken ships and document underwater cultural heritage. However, dealing with underwater environment encounters many difficulties such as the image degradation and blur, the light scattering, the reflections, the decrease of operating range in muddy and turbid waters, the lack of an accurate groundtruth etc. Hence, many image enhancements along with carefully system's design are yet to come.

One of the most important opportunities of Visual SLAM systems is the replacement of GPS in environments, where the GPS signal drops-off. Such environments are big cities, indoor scenes, underwater, tunnels, underground etc. SLAM is independent of satellite signal and uses an arbitrary coordinate system making accurate measurements of its physical surroundings.

There are various types of Visual SLAM systems with the two principal categories being direct and indirect methods. Direct methods based on a brightness consistency constraint, exploit information of the whole image being robust and resilient in texture-less environments. However, their high computational cost and their sensitivity to illumination changes make them less used by users and incapable for real-time performance. Conversely, indirect methods rely on salient feature detection and matching using a variety of detectors and descriptors (Rosten et al. 2006) such as FAST (Rosten et el., 2006), HARRIS (Harris et al. 1988), BRIEF (Calonder, 2010), ORB, SIFT (Lowe, 1999), SURF (Bay et al. 2006) etc. Feature-based algorithms make the transition from images to geometry easier and can allow wide-baseline matching using

invariant descriptors. Conversely, the generated 3D point cloud is sparse and usually colorless. Also, these algorithms do not sample across all available image data-edges and weak intensities. This diploma thesis focuses on the development and implementation of a real-time monocular Visual SLAM algorithm both for initialization of the map and localization in the same map. Nowadays, monocular Visual SLAM is of great significance, since a single camera will always be smaller, cheaper, more compact and easier to calibrate than a multi-camera system. However, it has its own limitations, as the depth cannot be recovered and hence, an accumulated drift is present. The drift occurs in 7 DoFs, i.e., translation, rotation and scale. This work is limited to an off-line process because the data used, had already been acquired. The performance of the feature-based algorithm, called ORB-SLAM 2 is conducted in two different environments, an underwater and an urban. The images in the underwater scenario were captured on-board ROV and showed an archaeological site of amphorae. The second dataset was acquired on-board MAV and depicted a part of Zürich in Switzerland. The results were compared with the groundtruth provided by their creators.
Furthermore, this thesis investigates the production of an incremental visual map with a subset of total images (keyframes), leveraging the feature extraction and matching thread of the examined algorithm. In this way, the visual map provides a higher accuracy and gives a more informative view of the environment to the user.

The system is not able to store/save or load the generated map except for camera's trajectory. The storage of the point cloud is essential for a post-processing and its load contributes to a second execution of the algorithm. Loading and localizing on a previously built map can improve the local accuracy. This thesis aims to provide such a map saving function which will export a PCD file once the procedure has stopped. This, in combination with the visual map can make ORB-SLAM's result more useful and comprehensible.

The thesis outline is presented below. Chapter 1 includes an extended literature review of existing Visual SLAM algorithms and some of their applications. Chapter 2 focuses on an in-depth analysis of the ORB-SLAM 2 system along with its preliminaries. Chapter 3 emphasizes on the development and implementation of this system as well as the difficulties that have been aroused. Finally, in Chapter 4 a brief conclusion is drawn based on the system's performance and its evaluation. Topics for future work are also presented.

2. State of the Art

2.1 Visual SLAM systems

SLAM is an abbreviation of Simultaneous Localization and Mapping which was firstly proposed by Durrant-Whyte et al. (1995). SLAM is a technique for obtaining the 3D reconstruction of an unknown environment in real-time, while estimating the sensor's motion in this environment. In recent years, SLAM has undergone a significant development, but it is still considered an emerging technology requiring further analysis and research. Visual SLAM is referring to those techniques which use a camera as the only sensor. In the literature, SLAM is abstracted into two categories, feature-based methods that consist of extracting salient features, and direct methods which optimize directly over pixel intensities instead of feature reprojection errors, allowing the use of whole information of an image. There are two types of feature-based methods, keyframe-based techniques, and filtering ones. Keyframe-based techniques rely on pulling out distinctive features from images called keyframes whereas filtering techniques use the whole information of an image with a probability distribution. According to Strasdat et al. (2012) keyframe approaches are more accurate than filtering methods due to computational cost. Over the years, many SLAM algorithms have been developed offering a powerful addition in the field of technology. Some of them are presented below.

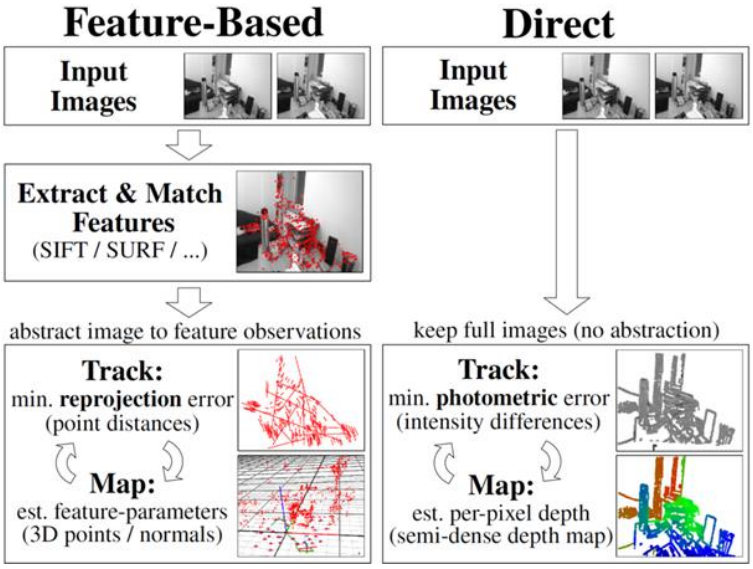


Figure 1. The general framework of feature based and direct SLAM methods (Durrant-Whyte et al. 2006).

2.2 Conventional Visual SLAM systems

2.2.1 Direct methods

DTAM (Dense Tracking and Mapping) presented by Newcombe et al. (2011), is a direct method where map initialization is performed using a stereo measurement. Tracking is conducted by comparing the input image with synthetic view images created from the built map. To enhance tracking, a pixel culling procedure takes place. Specifically, pixels whose photometric error is greater than a threshold are not considered. This error is derived for every combination of the reference image and any other image. It is used for estimating correspondences using the difference among pixel intensities. For distinctive pixels, the photometric error should be minimum. Camera pose is estimated using an alignment method between sequential frames to obtain rotational odometry providing flexibility to motion blur since successive frames have

similar blur. The mapping is done by leveraging multi-baseline stereo using the estimated camera poses.

LSD-SLAM (Large Scale Direct monocular SLAM) that was investigated by Engel et al. (2014), is a fully direct method where map initialization is achieved by using a first keyframe with random values set as depth for each pixel. To converge to a correct depth estimation later, a significant translation of the camera must be made. Tracking module includes the estimation of a relative pose between a new frame and a previous frame that was processed by minimizing the variance normalized photometric error. Also, LSD-SLAM takes into consideration the noise existing in depth calculations which varies from pixel to pixel depending on the distance from which they were observed. The next thread is depth map estimation where a keyframe selection is performed first. With the insertion of a new keyframe its depth map is computed by projecting all the points of the previous keyframe on it. After that, the depth map is scaled in such a way to have a mean inverse depth of one. The current keyframe is set as reference keyframe and is ready for tracking images. The pixels of its depth map can be optimized using all the frames that do not become keyframes. This is achieved by comparing small stereo baselines in regions where the stereo accuracy is expected to be large. The result of this is then integrated into the estimated depth map via a filtering approach. Then, a loop closure procedure that detects loops is performed and a reciprocal tracking test is applied for more consistency. Finally, map optimization is continuously executed with the aid of pose-graph (optimization). Nevertheless, this method is prone to scale-drift due to depth unawareness. For this reason, a direct scale-drift aware alignment is conducted and two keyframes with different scale can be aligned.

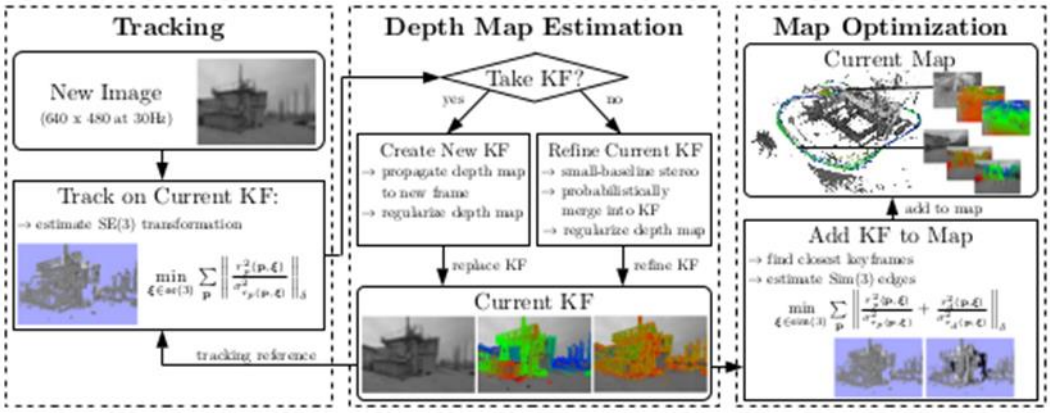


Figure 2. Overview of LSD-SLAM algorithm (Engel et al. 2014).

2.2.2 Indirect methods

A. Keyframe-based techniques

A representative and convincing keyframe method on which many algorithms were later based is Parallel Tracking and Mapping (PTAM). PTAM was introduced by Klein et al. (2007) and it can be summarized in the following steps: (i) the map is initialized using the 5-point-algorithm of Nister et al. (2004) from a stereo pair. This algorithm recovers the essential matrix (5 DoF-using 5 points) extracting the corresponding motion vectors of rotation and translation. The estimated relative pose of the stereo pair can then be used for the reconstruction of the initial map, (ii) new points are added, taking advantage of epipolar geometry (an epipolar search takes place in order to reduce the performing time), (iii) tracking and mapping are performed in two parallel threads so that tracking will not be affected by the computational cost of mapping. Tracking assumes that a 3D map has already been created while mapping counts on keyframes which are processed using Bundle Adjustment (Appendix). In this way, many points can be mapped. Also, one significant contribution of this method is the relocalization thread based on a feature tree classifier, which searches the nearest to the input frame, keyframe, allowing

recovery from tracking failure. However, apart from the need of human intervention for map bootstrapping, the system can fail in different ways such as tracking failure due to its dependence on corner features (implying that abrupt and rapid camera motions produce motion blur which eliminate the correct feature detection) and repeated structure generating a lot of outliers. Also, the mapping failure arises because of dysfunction of the 5-point-algorithm in the initialization procedure and perhaps a permanent change of real-world scene. Unfortunately, it has low invariance to viewpoint of the relocalization mode and lacks loop closing, factors that severely limit its application. The structure below summarizes the mapping thread's steps.

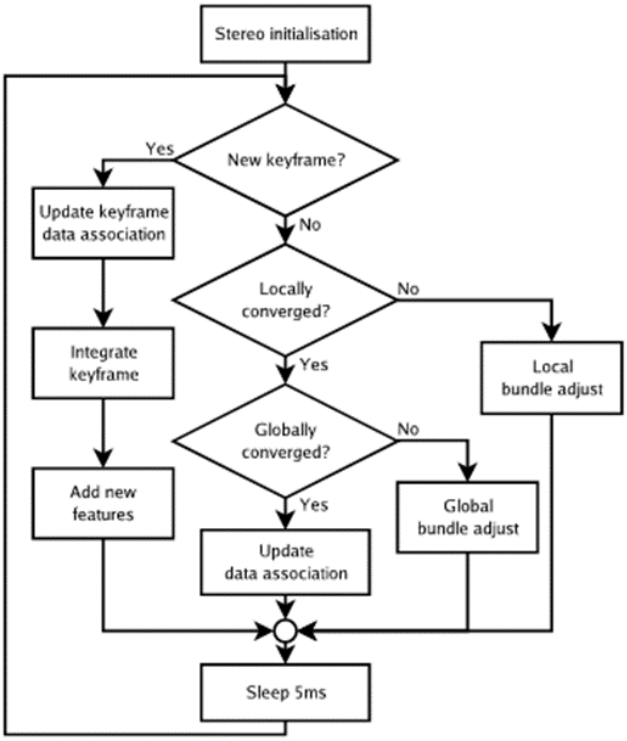
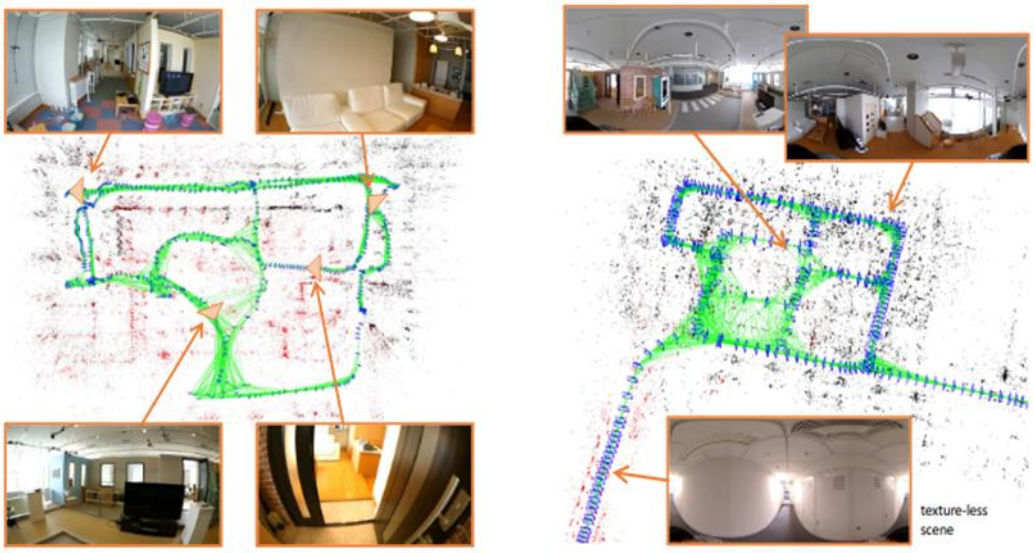


Figure 3. Mapping thread of PTAM (Klein G et al. 2007).

Open VSLAM established by Sumikura et al. (2019), is another keyframe-based method which introduces the implementation of new camera models. It can handle images from different types of cameras like perspective, fisheye and spherical. This software is divided into three parts: tracking, mapping and global optimization. In the tracking module, the pose of every frame is estimated via feature matching and can be optimized with the insertion of new frames seeing the same points. In this step, a keyframe selection is also performed. In mapping, new map points are triangulated using the keyframes mentioned above and a local Bundle Adjustment takes place. The last module consists of 3 steps: loop detection, pose-graph optimization, and global Bundle Adjustment. Loop detection searches for loops with every keyframe insertion with the purpose of correcting the drift accumulated: both sides are aligned, and duplicated map points are fused. This is achieved, using a pose-graph optimization which performs similarity transformations to fix the scale drift too.



(a) (b)

Figure 4. Mapping result: (a) using an indoor fish-eye video. (b) using an indoor equirectangular video (Sumikura et al. 2019).

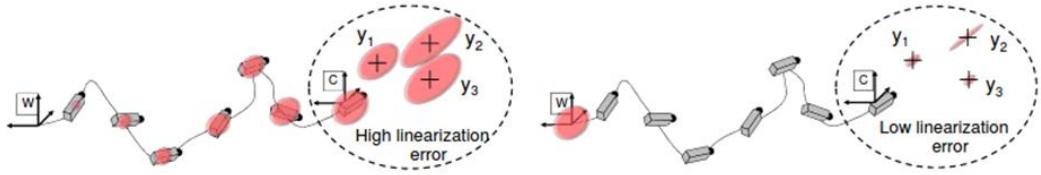
B. Filtering techniques

The first filter-based method is MonoSLAM introduced by Davison et al. (2007), which uses one camera as the only sensor. In this, camera pose, and 3D map points are calculated using the extended Kalman filter (EKF). The EKF is an extension of Kalman filter for non-linear systems where non-linearity is estimated using derivatives. In few words, Kalman filter is used when the interest variables cannot be measured directly, yet an indirect measurement is provided; and to calculate the best estimate of a variable (or state vectors) using several measurements from different sensors when noise is present. More specifically, a state vector is formed, consisting of the 6 degree of freedom camera motion (seven parameters are chosen to represent the position and orientation of the camera using a quaternion, i.e., an alternative way to represent a rotation around an axis in 3D space, which consists of 4 components, the x, y, z that stand for the rotation around the axis and w, for the number of rotations that will occur) and 3D position of feature points. This vector changes according to camera movement and so, new map points are added. It is worth noting that besides the state vector, a covariance matrix for map points is calculated. Some of the most important components of this method, are the use of a prediction model in the beginning and a feature point tracking result as an observation, and the performance of map initialization using known objects, providing features with known position and appearance. These features define the world coordinate system for SLAM allowing the scale knowledge. The main drawback of this method is the computational cost: The larger the environment, the larger the state vector becomes, making the real-time performance difficult.

A visual odometry algorithm that was investigated is 1-point RANSAC for EKF Filtering developed by Civera et al. (2010). It describes two main things. First, a six degree of freedom camera motion estimation is analyzed using the SfM method. A new technique which is based on high resolution images is presented. Secondly, a long-term robot trajectory is calculated using both monocular vision and wheel odometry.
1-point RANSAC: It uses one single feature correspondence for motion estimation leading to the lowest model parameterization. Given a set of points, it creates a random line using two of these points. It sets a threshold value t, to determine whether a point fits well to the model or not. More precisely, a buffer with size t is built up around the line. The points that are inside the buffer region are denoted as inliers, whereas the rest are outliers. Depending on the number

of iterations, the algorithm executes the above procedure with another starting point. Once all possible iterations have been performed, the one with most inliers will be selected. It is worth noting that the major difference from standard RANSAC is the number of points used for line construction resulting in the reduction of computational cost.

Camera-centered EKF: All feature positions and camera motions are referred to a frame that is local to the current camera unlike the standard EKF methods, where estimates are referred to a world reference frame. Therefore, linearization errors become smaller, resulting to lower uncertainties.



(a) (b)

Figure 5. (a) World-referenced EKF. (b) camera centered EKF estimation (Civera et al. 2009).

The implementation of this algorithm consists of a state vector which is formed by the location of the world reference frame x_W^{Ck} and the map of estimated features y^{Ck} both expressed in the current camera reference frame (Cadena et al., 2016).

$$x_k^{Ck} = \begin{pmatrix} x_W^{Ck} \\ y^{Ck} \end{pmatrix}$$

The 1-point RANSAC algorithm implemented here, takes advantage of the probabilistic prediction obtained from the EKF method to decrease the computational cost of the rejection step (discard spurious correspondences). Once the camera's calibration parameters are entered into the model, the algorithm starts processing the input image sequence. At the beginning, it selects a first image and searches for distinctive points. Then, it places these feature points in the second processing image. It creates a circle around each point with a radius equal to a threshold value set by the user and searches within it for the same feature of the previous image. If the located point in the second image is not the corresponding of the point on the previous image, the circle turns into an ellipse whose centers are these two points. Ellipses are colored red as long as searching points are within them. When the point is located on the line on which the algorithm searches for correspondences, the original red line becomes bold. When the point is as far away as the threshold value, the red line becomes thin. If a point is not detected within the set limits, the ellipse is colored purple. In case, the corresponding point does not exist in the image, the ellipse becomes blue and the algorithm stops the process.

Figure 6. Processing frame with circles and ellipses (Kourounioti 2018).

2.3 Deep Learning based Visual SLAM systems

Although SLAM techniques have brought rapid development in many tasks of computer vision, such as robotics, autonomous driving, 3D modeling, augmented reality etc., achieving high accuracy and performing in real-time, they must confront many difficulties such as the sensitivity of their performance in illumination, viewpoint changes and erratic camera movements. These problems are derived from the non-geometrical tasks of SLAM systems and they need a careful and fine-tuned system design so that they can be efficiently performed in different environments. Moreover, in visual odometry techniques, especially in monocular cases, a prior information must be known to retrieve the scale. In the last decades, Deep Learning (DL) has met a great evolution, focusing on recognition and classification tasks. Hence, the integration of DL techniques into visual SLAM systems is incredibly challenging, since the former rely on the extraction of appearance information of an image, limiting the performance of visual SLAM/odometry in new environments. Although DL techniques have not been leveraged enough by visual SLAM systems, several efforts have been made both on visual SLAM and visual odometry.

Wang et al. (2017) introduced a deep learning based visual odometry algorithm underlining the importance of geometric feature representation instead of image context to efficiently function within unexplored environments. They presented for the first time, a Recurrent Convolutional Neural Network (RCNN) which not only takes the above requirement into consideration but also, exploits the information taken by sequential images in the aim of estimating the motion between them. The proposed system takes as input a video or a set of consecutive images (monocular) which are barely modified by subtracting the mean RGB value of the training set from each pixel and resizing it in a multiple of 64. Two adjacent images are merged and then passed into the RCNN for feature extraction and sequential modeling of VO. In particular, the system consists of a CNN and a RNN which work jointly. Firstly, the image pair is imported to the CNN, which is composed of 9 convolutional layers, each of which is followed by a ReLU function except from the sixth one. The table below shows how the receptive field is progressively decreasing (for extracting exceedingly small feature points), the sizes of zero-paddings applied (to protect the dimensions of the image pair after performing convolution) and the increasing number of channels in the purpose of detecting multiple features.

layer	Receptive field	Padding	Strike	Number of Channels
Conv1	7x7	3	2	64
Conv2	5x5	2	2	128
Conv3	3x3	2	2	256
Conv3_1	3x3	1	1	256
Conv4	3x3	1	2	512
Conv4_1	3x3	1	1	512
Conv5	3x3	1	2	512
Conv5_1	3x3	1	1	512
Conv6	3x3	1	2	1024

Table 1. CNN configuration (Wang et al. 2017).

The CNN reduces the high dimensions of raw RGB images inserted at the beginning, facilitating the activation of the RNN. The product of the sixth convolution i.e., the learned feature is transferred into the RNN for sequential learning. The main benefit of RNN is that can efficiently model the motion relations between feature points by keeping memory of hidden states and performing loops i.e., the output of the current feature is “copied” and sent back as an input. Taking into consideration the previous capability, Wang et al., (2017) proposed the use of LSTMs, in order to manage long trajectories. In their study, two LSTMs are stacked together and each of them has 1000 hidden states. The output at each time step is a pose

estimation. Furthermore, they extend their work by calculating a pose uncertainty. After conducting some experiments, it is concluded that this DL enhanced VO system can compete the other state-of-the art conventional VO algorithms, estimating the absolute scale adequately, without any prior information. The structure below presents the RCNN system.

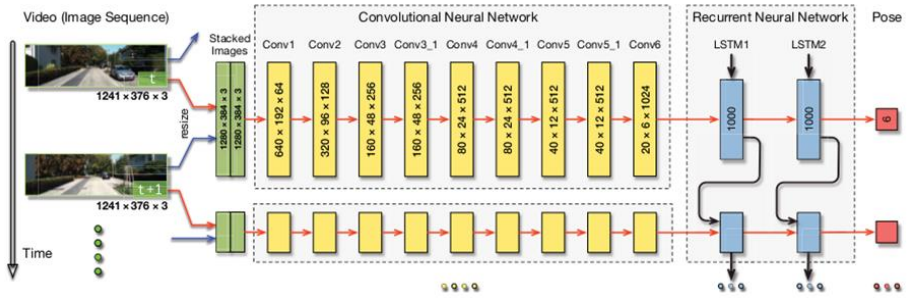


Figure 7. Illustration of the RCNN based VO system (Wang et al. 2017). images: KITTI dataset.

DF-SLAM (Deep Features based SLAM) system introduced by Kang et al. (2019) is a deep learning based visual SLAM system which relies on the extraction of deep feature descriptors acquired by a neural network. To maintain the efficiency of conventional features and perform in real-time, they proposed the creation of a shallow neural network producing local feature descriptors instead of replacing all the other traditional tasks. They kept the traditional pipeline of SLAM systems; the tracking, local mapping and loop closing threads. Tracking consists of localizing the camera using feature extraction. Moreover, it decides whether to insert a new keyframe. If tracking fails, global relocalization is executed leveraging the bag-of-words representation (DBoW) framework whose vocabulary is fed with the deep descriptors. Local mapping is responsible for the sparse 3d reconstruction of the environment. If a loop is detected, the loop closure thread will be activated, closing the loop, and performing global optimization. One of the most important tasks is feature matching, since all the threads mentioned above cannot process raw images. Kang et al. (2019) create a neural network exploiting the idea of triplets of Balntas et al. (2016) for its training. The CNN has two convolutional layers followed by Tanh function and a max pooling after the first convolution, to reduce certain parameters (speeding up the process). Finally, there is a fully connected layer which exports a 128d descriptor. An overview of the built system is presented below.

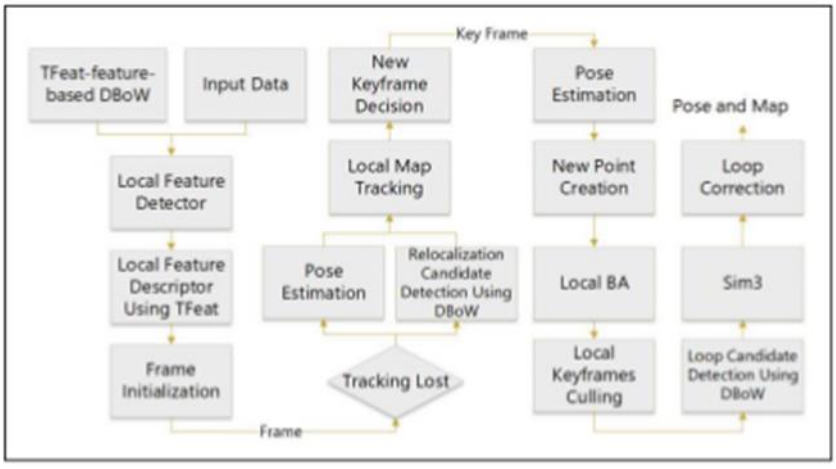


Figure 8. Framework of DF SLAM system (Kang et al. 2019).

The triplets mentioned above in combination with the work of Mishchuk et al., (2017) are formed as {a, p, n} [1], where a and p, are patches (positive) of the same physical point (i.e., different viewpoint) and n is a patch of a different keypoint (negative patch). “a” stands for anchor, “p” for positive and “n” for negative. Distances between them are computed but only

the two of them are used to not increase the computational cost. The hard negative is defined as the minimum distance between d (a, n) and d (p, n). In case that d (p, n) is smaller than d (a, n), the a and p, are swapped to {p, a, n} so that the 'hardest' negative is used for back-propagation. The loss function is formed as:

$$F = \frac{1}{N} \times \Sigma_{(0,N)}\, max(0,1 + d(a,p) - minimum\ distance)$$

Sheng et al. (2019) developed an unsupervised collaborative learning of a keyframe selection and Visual Odometry in monocular deep SLAM. They emphasized the importance of an efficient and strong keyframe selection task which can boost the localization and mapping threads; and then lead on a powerful optimization of the camera pose and depth prediction, yielding a potential visual odometry. Their deep model is formed by three networks; the keyframe selection network (which learns the similarity of the current frame and a keyframe, based on both visual and geometric characteristics), a camera motion network and a depth prediction one (which learn to predict the motions from adjacent frames and depth of the current image, respectively). Those three networks work collaboratively; namely, the similarity computed for keyframe selection relies on a geometric measurement arisen from the two latter networks and a visual network arisen from the keyframe selection sub-network. The visual odometry network consists of a monocular depth predictor (which takes into account multi-scale predictions to launch the local gradient) and a camera motion estimator (consisted of 8 convolutional layers followed by an average pooling) between a pair of frames. This deep learning-based SLAM system can detect and represent big geometric changes of the environment, since the motion estimator can use any image pair instead of sequential images only. The keyframe selection network contains a selector which constructs a set of keyframes by inserting frames that are geometrically and visually different from previous keyframes; and localizes frames to their closest existing keyframe. The selector has a two-stream structure. The selector combines the outputs from the visual and geometric stream and concludes to the final similarity score. The visual odometry network and the keyframe selection network are trained in different ways, both using a triplet of the form: $\{I_S, I_P, I_N\}$, where I_S is a small training sequence, where the center frame is the current frame, I_P is the temporally nearest keyframe and I_N is the nearest hard negative keyframe. During training, several modifications on images are made such as resizing and brightness. For each training, a loss function is calculated. The final loss function, a weighted combination of the previous losses is estimated in order to perform a global optimization.

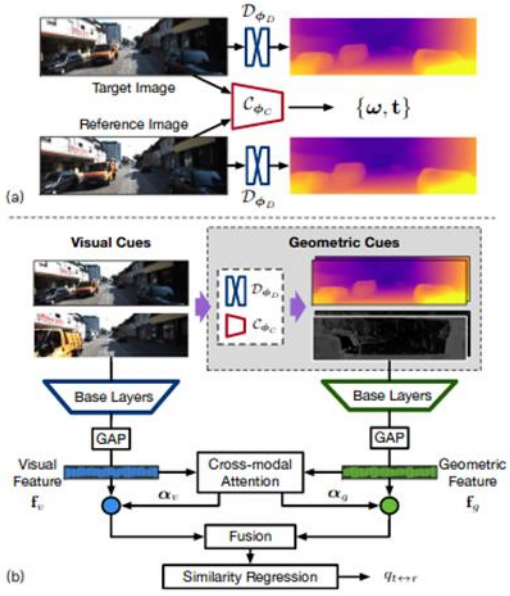


Figure 9. Structure of the three networks (Sheng et al. 2019).

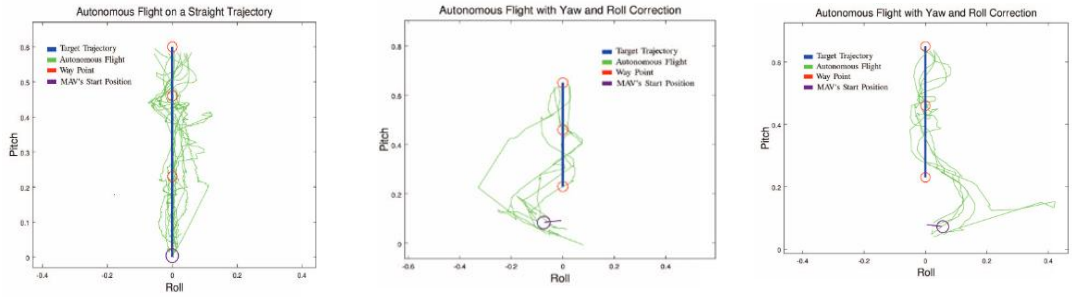
2.4 Applications of Visual SLAM systems/Visual Odometry

Kourounioti (2018) aims to produce orthophotos in two different approaches, evaluate them by comparing them with each other and implement a visual odometry method. Firstly, a dense colorless point cloud was created by the laser scanner ZEB-REVO which was used in backstreet of Kalamoti. This system is based on SLAM technique. After a proper editing and change of certain parameters, the mesh was produced. In order for the mesh to be colorized, another point cloud (dense and sparse) was built up using a multi-camera system consisting of a five-meter pole, on which two action cameras and a 360^{o} camera were placed at different heights. Also, a point cloud by a DSLR camera was used as reference for evaluating the accuracy of the laser scanner. These three point clouds were referred to the same coordinate system after the creation of a traverse including 15 survey stations and the use of two scanner spheres put on two surveying tripods at two of the traverse stations. More precisely, ground control points were placed throughout the area, on the facades of buildings and measured using a total station. Some salient features were also measured. With respect to the georeferenced of ZEB-REVO 's point cloud, the scanner spheres mentioned above were completely scanned as the laser scanner was moving forward. During the editing of point cloud, the sphere center's coordinates were imported from the traverse solving, adding the height from ground to sphere center. After the comparison between the above three point clouds one could conclude to the following:
- the point cloud produced by the DSLR was denser, uniform and covered the whole area since the images were of high resolution.
- the point cloud produced by the laser scanner lacked information at the higher parts of the buildings due to the steep angle of scanning and the narrow roads. As the incidence angle of the rays on a surface steepens, the number of the possible positions of a point becomes greater. A proposed but dangerous solution is the use of a pole on which the ZEB-REVO could be placed.
- the camera system encountered difficulty in regions where no points with known coordinates existed.

Afterwards, orthophotos were produced.
In the second part of this thesis, a visual odometry algorithm, the 1-point RANSAC for EKF Filtering by Civera et al., 2010 was implemented and evaluated to be leveraged in rapidly documentation conditions. Specifically, it can inform the user about regions that have been documented and the number of times that he has passed by. A GoPro Action Cam Hero 4 Black Adventure that was set to take 4K images was used. The experiment was conducted at an indoor corridor and two backstreets of Plaka, Athens (2-meter and 5-meter width). After the camera's calibration, frame extraction from a video and the change of their size in order to be compatible with the algorithm's requirements, the 1-point RANSAC for EKF Filtering was executed.
In view of the algorithm's methodology and the results of the previous implementation, one can draw several conclusions such as:
- The input image pairs must have a significant overlap making the algorithm easier to find features within the limits given and so not to terminate.
- The algorithm does not work well under texture-less surfaces since it cannot detect many points. Hence, the trajectory is estimated with low accuracy.
- The abrupt and quick camera movement hampers the feature extraction procedure.
- Feature points must be well-distributed all over the image.
- The algorithm is affected by the lens type e.g., normal, wide-angle. In this study, the algorithm found features only in the central part of images due to the very narrow roads and wide-angle lens usage. In other words, there was great radial distortion at the image's edges. However, when a normal lens is used, the algorithm tries to find features all over the image because of its limited field of view.
- As the threshold value becomes smaller, the algorithm finds more feature points (as it has less restrictions). However, it is assumed that the captured surface allows this finding.
- The measurements are heavily dependent on each other. Points that were not determined in correct position will be continuously appearing in wrong places throughout the path, as the current camera position is estimated considering its previous position only.

Martinez-Carranza et al. (2015) presented an implementation of autonomous flight in a straight trajectory, for a low-cost MAV, using ORB SLAM as a visual positioning system. For this, the Parrot AR Drone 2.0 was used, on which two cameras were incorporated: a horizontal and a vertical. The first camera (HD 720p) captured images at 30 fps, while the second one (320×240) captured at 60 fps. The system setup consisted of three main steps:
- a real-time video stream taken by MAV, was passed to the computer
- it was integrated into the ORB SLAM system, providing the quadcopter's positions and a 3D reconstructed map
- the control system is then activated, allowing for autonomous flight, leveraging the target trajectory

They used three waypoints that were recorded during the mapping thread as well as the starting point. Once the drone has reached the last waypoint, which signaled the end of the flight, it lands autonomously. After the first flight (target trajectory was generated), they performed three different flights. In particular, they put the vehicle in the starting position in three different ways; First, it was looking forward in the line determined by the target trajectory (yaw value was approximately zero), secondly, it was placed on the left of the target trajectory and thirdly, it was placed on the right. In the two latter conditions, the vehicle was also placed a bit further from the starting point. Therefore, the controller had to rotate the drone to move on the line defined by the target trajectory. Immediately after taking off, the control system computed the control signals using drone's positions. The main purpose of the controller was to pilot the vehicle in such way that it followed the straight line. Every time that the MAV reached a waypoint, it hovered for a few seconds and it continued its route towards the next waypoint. In order for the system to realize that the drone had reached a waypoint, a radius around it, was set and the value of pitch angle should be inside this circle. According to the results, Martinez-Carranza et al. (2015) concluded that there was a delay in receiving images from MAV which caused the control system to use a previous position. Hence, the vehicle was noticed by the controller a few milliseconds later after passing the target point. This affected the precision of autonomous navigation whose average pitch error was 2% with reference to the total length of the target trajectory (six meters in real world and 0.6 according to ORB SLAM units) while the average roll error was 8%. Despite those difficulties, the vehicle could be efficiently piloted by the control system, passing by the waypoints, and achieving a landing at the final position.



(a) (b) (c)

Figure 10. Autonomous flight: (a) On a straight trajectory. (b) vehicle located on the left-with yaw and roll change. (c) vehicle located on the right- with yaw and roll change (Martínez-Carranza et al. 2015).

Another implementation was conducted by Handa et al. (2014) who applied RGB-D Visual Odometry and SLAM techniques for the 3D reconstruction of certain environments. Specifically, they used a handheld camera capturing RGB-D images of a living room and an office room. Those images contributed to the quantitative evaluation of the model's accuracy by providing information for the user's path during image acquisition. At first, Handa et al. (2014) calibrated the camera achieving lens distortion elimination from all the images. In every room, they were taking images in four different trajectories. The experiments were classified in two different settings; in the first condition, data were considered to be perfect without noise whereas the second condition assumed real data with noise existed in both RGB and depth

images since there was intense illumination from different sources. After the mesh model creating, the surfaces were georeferenced in the same system and the models were compared to each other. Subsequently, each trajectory of every room was exported for all conditions. The results demonstrated that it was feasible to efficiently estimate the camera trajectories, compare the models with each other and deduce the quality of the model's surface for every situation.

Ortiz-Coder et al. (2019) presented and tested a portable mobile mapping system for the 3D reconstruction of the Roman Aqueduct of Miracles in the city of Merida (Spain). The platform consisted of two cameras A and B, located in such way that their optical axes were parallel. The aim of their work was the development of a fully automatic system with continuous data capture without any human intervention. They performed three experiments where the user was moving in a perpendicular direction to the camera optical axis: three data capture scenarios at different observation distances between the system and the monument's base (5, 12 and 20 meters). The workflow of their proposed system included four main processes:
(i) Both cameras started the video capture at 25 and 4 fps. The ORB SLAM algorithm was implemented in images captured from camera A, providing their positions (a text file with the trajectory). This step assumed that a camera calibration was first conducted.
(ii) Some filters were applied in images captured from camera B, removing frames that had small baseline, that were taken when the camera was stable or had done a small movement, that were redundant (comparing with the images of A) and frames that had angles (ω and k) that were rotated at around 9^{o}. For the determination of redundant frames, both cameras were synchronized. For the synchronization (parameters that related the position of one camera with the other), they measured with a total station 15 target points in two walls of the monument to be used into the transformation estimation.
(iii) A segmentation procedure was then performed, grouping three consecutive images into a "segment" that had several correspondences among them.
(iv) The colored point cloud was generated using an open-source photogrammetric software.

Ultimately, Ortiz-Coder et al. (2019) compared the results (based on RMSE of each direction) with a usual photogrammetric process using a reflex camera and the Agisoft Metashape for the point cloud creation. To evaluate their accuracy, target points (40) were measured with a total station in a local coordinate system. Those points were used as reference points. The images were taken from the same trajectories followed by the mobile platform and at the same distances. They concluded that their system can effectively compete other classical photogrammetric processes in terms of accuracy and time, since the average error and processing time (for distances of 5 and 12 meters) were only a bit higher in the proposed approach and the RMSE was a bit lower.

3. Theoretical Approach-Preliminaries

3.1 Visual SLAM Framework

SLAM can be described as a chicken-and-egg problem, meaning that the position and orientation of sensor motion must be known in order to accurately reconstruct an unknown environment. However, the estimation of the sensor motion needs the information of map points of the environment, i.e., 3D position. With the availability of low-cost sensors and fast improvement of technology, SLAM techniques have come a long way in the past few years, being widely proposed in many fields such as computer vision, robotics, AR, etc. In case, the sensor is a camera, it is referred to as visual SLAM, as it is based on visual information only. Visual SLAM has much more limitations compared with other sensor-based SLAM systems (360 laser sensing) due to less visual input acquisition when the field of view is limited. However, many different types of sensors were integrated in the already existing algorithms like laser range finders, inertial sensors, and GPS. When SLAM systems use a single camera as the only sensor, it is called monocular SLAM, something particularly challenging for depth definition which is derived from multi view geometry.
A first solution of SLAM problem was introduced by Durrant et al. (2006). In their work, it was highlighted that the observation of feature points in the environment and the correlation of their estimated positions could contribute to the calculation of the observer's pose and its orientation. The true positions are never known or measured directly.

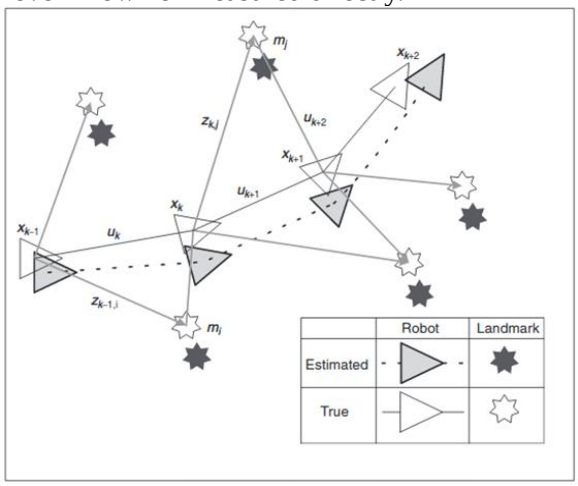


Figure 11. The essential SLAM problem (Durrant-Whyte et al. 2006).

Formulation of Visual SLAM:

In general, SLAM is a process where a moving autonomous mobile robot reconstructs a 3D map of the unknown environment while it estimates its pose using this map. In other words, the robot's trajectory and all landmarks' positions are calculated online and simultaneously. Assuming that a sensor is located on the mobile robot and it observes several landmarks in the environment while moving, the following are defined for a time k:

- x_k is the vector that is describing the robot's position and orientation.
- u_k is the command executed by the robot at time k-1 that makes it move to x_k position.
- m_k is a vector of a landmark's position in the world.
- z_{ik} is a robot's observation of a landmark i, at time k.

The figure VII represents the SLAM problem for a robot moving through the environment (white arrows). While robot moves, it has sensors, to perceive where the landmarks (white) are in the environment with respect to its position. In reality, these observations are noisy as well as the robot's motions, leading to an uncertainty. Therefore, the system believes that is actually

moving on the gray arrows based on e.g., wheel encoders. Now, it perceives the black landmarks instead of the real ones. In this way, the real map and the map generated from the platform are not identical as well as the estimated trajectory since it is based on the landmarks' observation. The offset between white and black landmarks reflects the existing error. This error can be corrected by robot's returning to a previously revisited area.

The latest visual SLAM systems work under this framework. They are composed of three basic modules: initialization, tracking and mapping. After the definition of a certain coordinate system, a part of the unexplored environment is reconstructed, referred to as initial map. After that, tracking is enabled, meaning that the reconstructed map is tracked in the image and the camera pose is computed by solving a PnP problem (Appendix). When the camera observes regions that have never been discovered before, the mapping procedure is performed by expanding the already built map. Many modern algorithms incorporate two additional processes, i.e. relocalization and loop closure detection allowing life-long operations. Relocalization helps the system to automatically recover from tracking failure, caused by blur, erratic camera motions and occlusions, maintaining the consistency of the map. Loop closure detection improve the SLAM system in a manner that it can recognize when it has returned to a previously mapped area. The classic problem is that two places in the reconstructed map are turned out to be the same place in the world, even though their true positions are different. Thus, the system should calculate the transformation, to align the "two parts", i.e., close the loop. Generally, SLAM techniques focus on minimizing the reprojection error, usually through Bundle Adjustment (Appendix). The main purpose of visual SLAM is the real-time performance, so in most cases, the optimization of both camera poses, and map points is achieved separately but at the same time before they finally get merged.

3.2 Visual Odometry Framework

Visual Odometry (VO) – camera-based odometry-focuses on estimating the changes of camera motions over time and the 3D structure of an unknown environment, in real-time. It was first introduced by Nister et al. (2004). Odometry and visual SLAM are similar techniques because both are responsible for camera position estimation. The main difference between them is the map optimization in the mapping procedure. In VO, the map coherence depends on a small part of the reconstructed map, potentially optimized only over the last n poses (Scaramuzza et al. 2011) by minimizing the sum of squared reprojection errors of reconstructed map points, which is called windowed bundle adjustment (local refinement). Conversely, visual SLAM considers the whole map, performing global optimization. The relationship between them can be represented as:

Visual SLAM = Visual Odometry + Map Optimization + **(Loop closure)**

It is worth noting that visual SLAM is more precise than VO, as it integrates much more constraints in the path estimate but not necessarily more robust, because the possible existence of outliers in loop closing procedure can negatively affect the consistency of the map created (Scaramuzza et al. 2011). According to Cadena et al. (2016), VO systems interpret the environment as an endless corridor, whereas SLAM algorithms understand the real topology of the world due to loop closing techniques and therefore, they are able to discover "shortcuts" in the map.

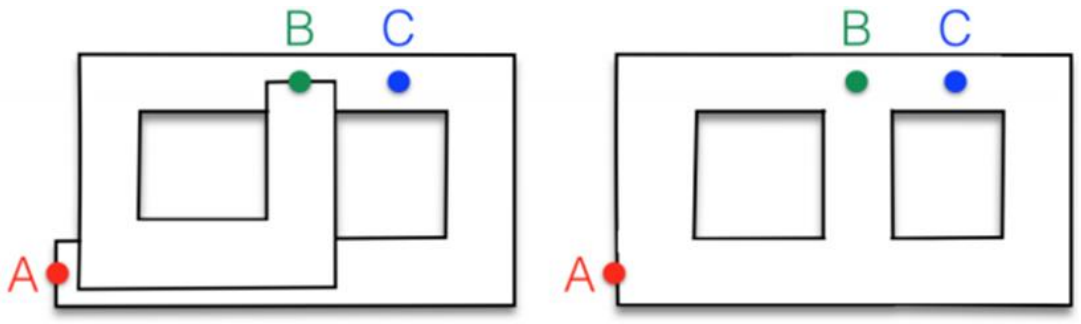


Figure 12. Left: Map built using VO. Right: Map built using SLAM techniques (Chotrov et al. 2018).

Formulation of Visual Odometry:
Assume that someone is walking down in an unknown environment and taking shots at distinct time instants **n** with a camera system attached. Two camera positions at two different time instants **n** and **n-1** can be linked through a rigid body transformation:

$$T_{n,n-1} = \begin{bmatrix} R_{n.n-1} & t_{n,n-1} \\ 0 & 1 \end{bmatrix}$$

, where $t_{n,n-1}$ is the translation vector and Rn, n-1 is the rotation matrix. $T_{n,\,n-1}$ contains all the camera motions. To compute the current camera pose C_n, a concatenation of all transformations T must be done. Hence, $C_{n-1} = C_{n-1} * T_n$.
The camera pose C_0 for the instance n=0, can be randomly set by the user.
In a nutshell, visual odometry's framework can be summarized in the following main steps:
sequential images → feature detection → feature matching → camera motion estimate → windowed bundle adjustment (local refinement)

3.3 Structure from Motion-SfM Framework

SfM is the process of reconstructing the 3D structure of an object from its projection into a set of sequential images taken from different viewpoints. SfM has met a significant evolution over the years, thus increasingly large-scale reconstruction systems have begun to be built up. Existing SfM methods can be divided into two categories: the incremental structure from motion and the global SfM. The former functions with a set of unordered images and performs local optimization of both camera poses and map points (Local Bundle Adjustment) after the insertion of k new images (k is a specified value) to mitigate the wrong connections among images. Conversely, global SfM performs global Bundle Adjustment (BA) once, at the end of the reconstruction and so, it is more susceptible to outliers. Hence, incremental BA is slower but more robust and accurate than global BA. Nevertheless, incremental BA assumes that a pair of two correspondences with large intersection angle and an adequate number of well-spread matches are selected at the beginning. Incremental Bundle Adjustment's workflow consists of two main parts; the termed correspondence search and incremental reconstruction (Jiang et al. 2020).

Termed Correspondence search:
1. Feature extraction: SfM starts by detecting discriminative keypoints in each image, defining a region around them, normalizing the content of this region, and computing a local descriptor for each of them, using several detectors and descriptors such as SIFT (Lowe, 1999), SURF (Bay et al. 2006), FAST (Rosten et el., 2006), BRIEF (Calonder et al., 2010), ORB (Rublee et al., 2011) etc. The features should be invariant under geometric and radiometric changes, so that can be detected by SfM in different images.
2. Feature matching: is the part of determining correspondences between images that observe the same place. SfM detects those images by exploiting the information of descriptors. For this, a similarity measure applied to feature descriptors, is used to find the most similar keypoints.

In the case of binary descriptors, the measure is based on the Hamming distance, otherwise the Euclidean distance is computed. Feature matching, exploiting the temporal consistency constraint can be limited to their forward and backward neighbors within a specified threshold (Schonberger, 2014).

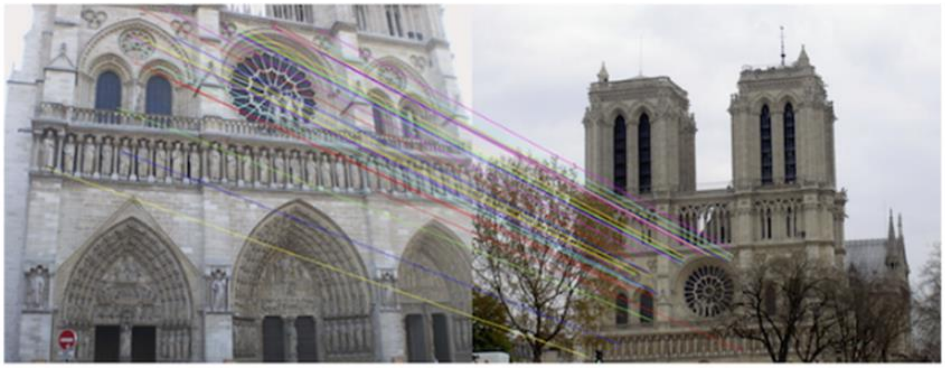


Figure 13. Two images from the Notre-Dame Cathedral with corresponding feature points using SIFT algorithm (Schonberger et al. 2016).

3. Geometrical verification: Once the initial matches have been found, a geometric verification is applied in order to remove the noise existing in the correspondences, since the latter were based entirely on the appearance (descriptors). The procedure consists of two steps. Firstly, local photometric and geometrical constraints are applied in purpose of reducing the number of outliers and thus, increasing the inlier ratio of matches (Jiang S. et al. 2020). Secondly, leveraging the epipolar geometry, an essential (5-point algorithm) and a fundamental matrix (7-point algorithm) are calculated for calibrated and uncalibrated cameras, respectively, inside a RANSAC scheme to refine the remaining correspondences. This makes the feature matching task more robust and reliable.

Incremental reconstruction:

1.Initialization: SfM consciously chooses a two-view reconstruction. More precisely, it selects an initial pair of images that have much overlap with several other cameras (i.e., dense seed location) and satisfy some constraints such as, the wide baseline and the well-distribution of keypoints. This would lead to a more powerful and accurate reconstruction whereas an initialization that does not fulfill the previous requirements and derives from a sparser image location would result in a faster but less accurate performance, thus it sacrifices the accuracy for time speed. Initialization is a crucial part in SfM workflow since a reconstruction may never be retrieved from a bad initialization.

2. Image registration: Once an initial model has been reconstructed, new images can be added by solving a PnP problem. The camera pose and the intrinsic parameters in the case of uncalibrated cameras, can be estimated using the 2D-3D correspondences between the current image and the already processed images, that have triangulated points associated with them. However, the correspondences contain many outliers and thus, the solution needs refinement. This is achieved using either the RANSAC algorithm or other minimal solvers (Schonberger, 2016).

3. Triangulation: Newly added images should observe the already mapped points. Additionally, new points can be triangulated if there is at least one more image that sees them from a different viewpoint. This step is vital for the SfM procedure as it expands the reconstructed model and allows the registration of new images offering more 2D-3D correspondences.

4. Bundle Adjustment: the previous two procedures are highly connected, as the uncertainty of camera poses is transferred to triangulated points and vice versa. However, new triangulations improve the camera pose estimation. To subsequently optimize both the camera positions and the mapped points, bundle adjustment is conducted, minimizing the reprojection error (Appendix). With the insertion of a new image, the local BA is performed in the aim of mitigating the errors occurred. Finally, a global BA is executed after all images being registered,

guaranteeing the consistency of the reconstructed scene. Although, SfM provides high accuracy due to its iterative local BA, it is limited when a large number of images should be processed.

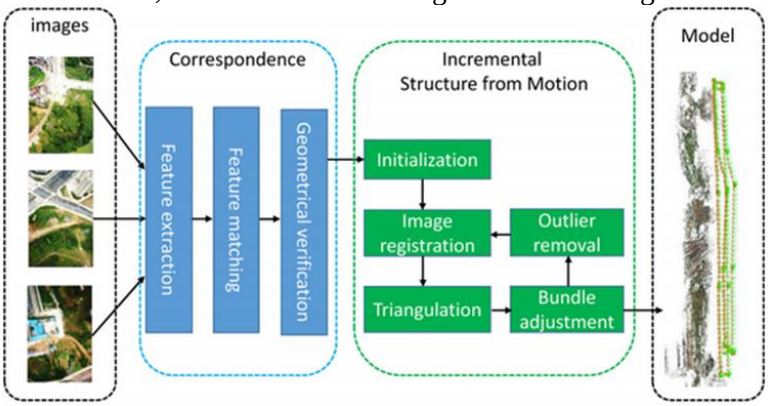


Figure 14. Pipeline of incremental SfM (Jiang et al. 2020).

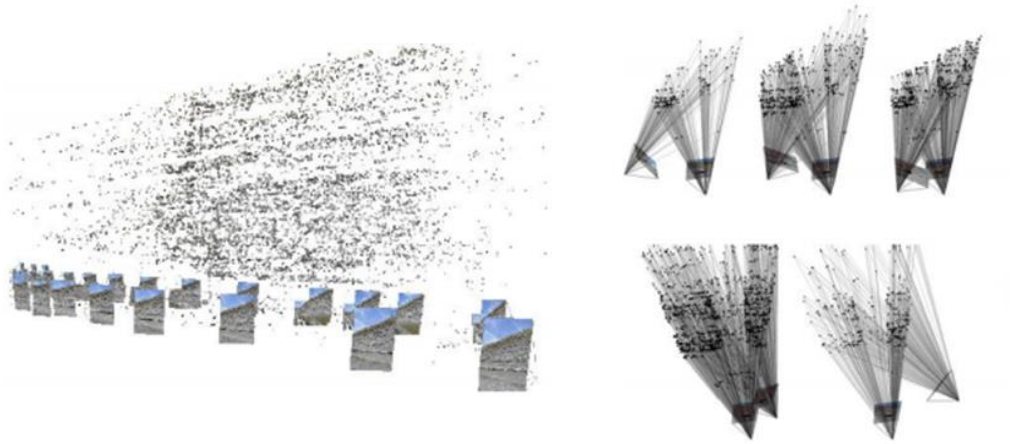


Figure 15. Left: Relative orientation results with respect to one key-frame. Right: sparse point cloud and camera positions using 72 images (Thoeni et al. 2012).

Epipolar Geometry:

Epipolar geometry is the geometry of stereo vision (Epipolar geometry, Wikipedia). Assume that there are two cameras capturing a scene in 3D real world, e.g., P point, from two different positions. The aim of epipolar geometry is finding the corresponding point p_R (projection of point P) in the second image plane, given its position in the first image, p_L. Therefore, the standard epipolar geometry is illustrated by the two cameras observing the same point in real world P, whose projection is p_L and p_R at first and second image, respectively and the two camera centers C_L and C_R. The figure below shows the relations between the above terms:

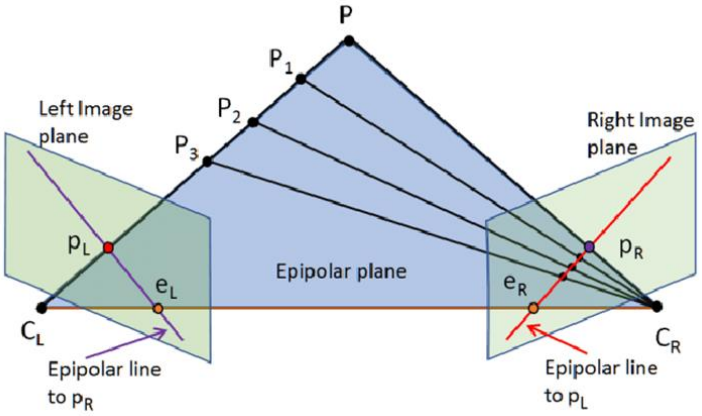


Figure 16. Epipolar geometry (Cadena et al. 2016).

Epipolar plane is the plane defined by C_L and C_R and point P. So, epipolar plane depends on capturing points, as different points generate different epipolar planes.

Epipole (e_L, e_R) is the projection of the other camera's projection center into the image.

Epipolar line is the intersection of epipolar plane and image plane (connects the epipole with the corresponding point).

Epipole axis is the line connecting the two projection centers. Hence, epipolar planes can be rotated around it.

It can be concluded from the foregoing that epipolar geometry allows an efficient and "predicted" search of corresponding points and the reduction of search space from 2D to 1D, making data association faster. For example, assume that there is an image with multiple objects which look very similar to each other and a corresponding point in this image. Looking for the same point in the second image may be difficult due to the similar feature descriptors the points would have. The restriction of the search from the whole image to a single line enables the reduction of search space mentioned above. Also, epipolar geometry can limit the possible mistakes that can be made during data association.

RANSAC – Random Sampling Consensus:

The RANSAC algorithm, proposed by Fischler & Bolles (1981) at SRI International, is an iterative method and one of most popular tools for estimating parameters of a mathematical model given a set of observed data which contains outliers. It is a non-deterministic technique in the sense that it generates a sensible result with a certain probability. This probability becomes bigger as more iterations are allowed. It can also be described as an outlier detection method (Random Sampling Consensus, Wikipedia). The algorithm consists of the following main steps (Fischler et al. 1981):

1. It randomly selects the minimum number of matched points required to determine the model's parameters, i.e., to estimate the transformation by calculating three points for a six-degree-of-freedom camera system.
2. It estimates the rotation and translation parameters using the selected points.
3. The remaining points are transformed according to the previous transformation.
4. It calculates the distance between new transformed points and their correspondences.
5. Corresponding points whose distance is less than a threshold value set by the user are considered as inliers.
6. It repeats the above steps k times (k is defined by the user).
7. Iteration with most inliers is considered to be the "best".
8. Refine the "best" transformation using least squares on the inliers.

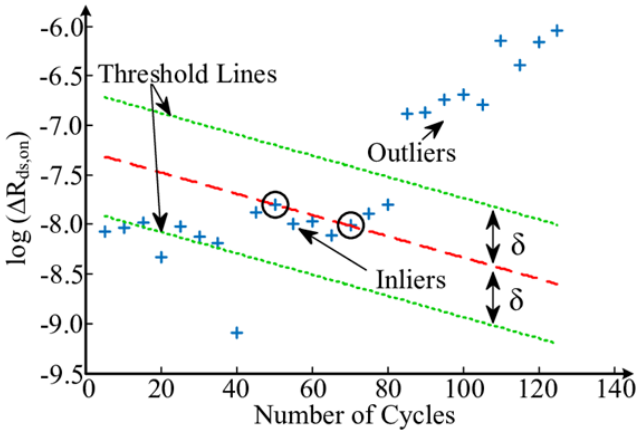


Figure 17. Illustration of the threshold value determined by RANSAC algorithm to detect outliers (Dusmez et al. 2017).

Camera Calibration:

Camera calibration or geometric camera calibration is crucial in 3D computer vision to obtain metric information from 2D images. It is the process of estimating the parameters of the intrinsic orientation that fits better to the model examined "camera-lens". The intrinsic parameters describing the physical camera model, include the coordinates of the principal point (x_0, y_0), the focal length that is represented by (a_x, a_y) in terms of pixels, the skew coefficient (γ) between the x and y axis, which is often 0, the coefficients of radial distortion (k_1, k_2, k_3 etc.) in the case of a non-pinhole camera and the different x and y scale factors. The result of camera calibration is a 3×4 matrix, known as camera matrix K, which only contains the linear parameters.

$$K = \begin{bmatrix} a_x & \gamma & x_0 & 0 \\ 0 & a_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The non-linear parameters like the coefficients of radial distortion cannot be taken into consideration in the linear camera model. They can be computed during the non-linear optimization techniques, e.g., bundle adjustment. One of the most common methods for camera calibration is Zhang's technique (Zhang, 2000). It requires the camera to observe a planar pattern which should be captured from at least, two different orientations. However, four or five orientations are recommended for achieving better quality. Both the camera and the pattern can be randomly moved. Zhang's method calculates the intrinsic as well as the extrinsic parameters. The extrinsic parameters determine the position and orientation of the camera with respect to a world reference frame (i.e., relative translation and orientation). For distortions to be considered, the solution of the simple linear problem is leveraged and inserted into a non-linear minimization algorithm. In particular, the latter optimizes the first solution based on a maximum likelihood criterion.

3.4 Description of ORB-SLAM 2

Why ORB-SLAM?

In recent years, visual SLAM systems build maps that are used merely, to estimate the camera pose during a certain session. Therefore, they are not designed to localize a different camera or to relocalize the same one from other viewpoints. Also, they cannot handle non static scenarios. So, maps need to be recognized to overcome the above limitation.
For this reason, Mur-Artal et al. (2017) introduced a new feature-based monocular, stereo, and RGB-D SLAM system, the ORB-SLAM 2 (is named after Oriented FAST detector and Rotated Brief descriptor) which can operate in real time, in a wide variety of environments like small and large-scale indoor and outdoor environments handling hand-held, drone-flying, industrial and car-driving sequences. It is worth noting that the algorithm uses the same feature points for all tasks, i.e., tracking, mapping, relocalization and loop closing, leading to the reduction of computational cost, and allowing an efficient and reliable process. It uses ORB features-oriented multiscale FAST corners (Rosten et el., 2006) with 256-bit descriptor BRIEF (Calonder et al., 2010) associated-which enable a real-time performance without GPUs, offering good invariance to illumination and viewpoint changes. This allows the matching with wide baselines enhancing the Bundle Adjustment's accuracy. In particular, as the baseline becomes bigger, the angle between the world map point and its projections in two images becomes larger and thus, it provides a good stereo observation. The system can be summarized to the following main threads: tracking, local mapping and loop closing that are executed in parallel, and global Bundle Adjustment.

Map Initialization

ORB-SLAM 2's map initialization is a fully automatic procedure that is independent of the scene. It aims to estimate the relative pose of two sequential frames to create an initial map of planar and non-planar scenes. First, an estimation of two geometrical models (a homography and a fundamental matrix) is performed. Then, a heuristic method takes place for the model selection bearing in mind the need of a configuration with significant parallax and resulting in the creation of a non-corrupted map. The following steps describe the algorithm:
1. It extracts ORB features in the current frame F_C and searches for correspondences in the reference F_R frame. In case that not enough matches are found, it selects another reference frame. Correspondences are described as x_C and x_R.
2. It calculates at the same time and iteratively a homography H_{CR} for planar scenes and a fundamental matrix F_{CR} for non-planar scenes using the direct linear transformation and eight-point algorithms respectively with RANSAC scheme (the homography is a special case of fundamental matrix).

$$x_C = H_{CR} * X_R \quad (1)$$

$$x_C^T * F_{CR} * x_R \quad (2)$$

For both models, the same number of iterations is selected and the points that are used at each iteration are: four for the homography and eight for the fundamental matrix. More specifically, at each iteration a score S_M for each model (M) is computed as follows:

$$S_M = \sum[\rho_M * (d_{CR}^2 * (x_C^i, x_R^i, \mathrm{M})) + \rho_M * (d_{RC}^2 * (x_C^i, x_R^i, \mathrm{M}))]$$

, where $\rho_M(d^2) = \begin{cases} \Gamma - d^2, & d^2 < T_M \\ 0, & d^2 \geq T_M \end{cases}$

d_{CR}^2 and d_{RC}^2 are the symmetric transfer errors from one frame to the other.

T_M is the outlier rejection threshold based on the χ^2 test for confidence level of 95%.. Specifically, T_H and T_F equal to 5.99 and 3.84 respectively, assuming a standard deviation of one pixel in the measurement errors (Raul Mur-Artal et al. 2017).

Γ is equal to T_H so that both models score equally for the same d in their inlier region.

The homography (1) and the fundamental matrix (2) with the highest score are selected. The process is restarted if no model could be found.

In the case of stereo or RGB-D SLAM, where depth can be obtained from only one frame, the first processing frame is selected as a keyframe. Its pose is set to the origin and a map is being created from stereo keypoints.

3. It selects the model that fit better according to the surface. In general, a planar or nearly planar scene or surface with low parallax can be effectively explained by a homography. However, a non-planar surface with sufficient parallax can be described by the fundamental matrix. In both cases, the other model can be selected but the problem will not be well constrained (Hartley et al. 2003). It calculates $\boldsymbol{R_H = S_H / (S_H + S_F)}$, using the previous estimated scores. If R_H is bigger than 0.45, the homography is selected to describe the current surface, otherwise the fundamental matrix is chosen.

4. To recover the best motion hypothesis, a triangulation of the eight and four solutions in the case of the homography and fundamental matrix, respectively, is conducted. After the triangulation, the algorithm searches for one unique solution with most points seen with parallax, in front of both cameras and with low reprojection error (Mur-Artal 2017). In the case of a non-planar scene, the fundamental matrix is converted into an essential matrix through calibration matrix K as follows:

$$E_{RC} = K^T * F_{RC} * K$$

If there is not a clear solution, the procedure starts from step 1.

5. Assume that the motion has been retrieved, a full Bundle Adjustment is performed to optimize the initial reconstructed map.

The ORB-SLAM' s 2 coordinate system is defined as:

- Origin: the first processed image
- units: the relative pose of the first two frames
- scale: initial map's median depth

<u>Tracking</u>

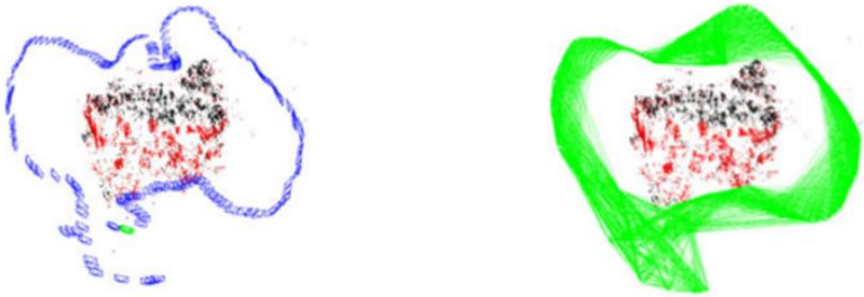
Once an initial map exists, the tracking thread calculates the camera pose with every incoming frame (Mur-Artal et al. 2014). Monocular tracking consists of five important steps which are presented below:

1. First, the ORB extraction is performed. In detail, the system computes a scale image pyramid and then it extracts FAST corners at each level. To increase the keypoint distribution, the frame is discretized in a grid retaining the same number of features in each cell (Mur-Artal et al. 2014). In the case of low contrast or textureless surfaces, there is a possibility that some cells may not include corners. Therefore, the number of corners maintained per cell should be adapted, meaning that the rest of cells are allowed to keep more keypoints. The parameters involved in

this step can be modified according to the environment. Mur-Artal et al. (2017) proposed the extraction of 1000 keypoints at 8 pyramid scales with a scale factor of 1.2 for image resolutions from 512×384 to 752×480 and 2000 keypoints for higher resolutions.
2. Secondly, the pose of the current frame is estimated using the previous frame (assume that tracking was successfully performed). For each ORB feature of the previous frame which has a map point associated, the algorithm searches for correspondences in the current image in a small area around its previous position (Mur-Artal et al. 2014). If it cannot find enough matches, the area becomes larger. The matches are optimized by an orientation consistency test (Mur-Artal et al. 2014). Finally, the camera pose can be calculated using these 3D (map points of previous frame)-2D (matches in current frame) correspondences solving a perspective-n-point problem with RANSAC.
(3). In case that tracking in previous processing frame was lost (that can be caused by occlusions or abrupt camera movements), the relocalization thread (locating the camera on an existed map) takes place. More precisely, the frame is converted into its bag-of-words representation (Mur-Artal et al. 2014). The bag-of-words representation is a summary of an image using visual words. The ORB- SLAM 2 relies on the DBoW2 recognizer, which is described below. The system queries the keyframe database for candidates (keyframes that have similar appearance to the current one). As the processing keyframes are sequential and there is an overlap between them, there will not be a singular keyframe candidate with a high score. The recognition procedure returns all keyframe candidates whose scores are higher than the 75% of the best score. Afterwards, the algorithm searches for ORB matches between the current frame and the candidates, in which ORB keypoints are associated with map points. Hence, there is a set of 2D to 3D correspondences for each candidate. To compute the current frame's camera pose, a PnP problem at each iteration is solved inside a RANSAC scheme for each candidate. With the aim of accelerating the search procedure, D. Galvez-López et al. (2012) proposed to limit the matching to those features which belong to same node at the second level of the vocabulary tree (which will be analyzed below). These matches are refined by an orientation consistency test which rejects outliers, enhancing their consistent rotation.
4. Once a camera pose with most inliers is retrieved, a guided search of more correspondences with the map points is conducted as described below. A local reconstructed map is projected into the frame instead of the whole map, reducing the perplexity of large environments. This local map contains a set of keyframes K_1 that share map points with the current frame and a set K_2 which are neighbors to the K_1 in the Covisibility graph. In the case of very interconnected maps, the set K_2 may be extremely big. Therefore, the number of neighbor keyframes is reduced. Also, there is a reference keyframe K_{REF} which belongs to K_1 set and has most map points with the current frame. Correspondences between map points, which are seen in the K_1 and K_2 keyframes, and points in the current frame are found as follows:
- The map point is projected into the frame, x and it is discarded if it lays out of it.
- The angle between the viewing ray **v** and map point's patch normal *n* is estimated.
If $v \cdot n < \cos(45^o)$, the map point is discarded. *n* is the mean vector of the viewing directions (Mur-Artal R. et al. 2014).
- The distance **d** between the map point and camera center is computed. If *d* is not included in $[d_{min}, d_{max}]$, i.e., scale invariance region, the map point is discarded.
$d_{min}$and d_{max} are distances at which the map point can be observed, according to the scale and distance at which it has been seen (Mur-Artal R. et al., 2014).
- The scale is calculated by the ratio d/d_{min}.
- The descriptors of unmatched feature points near x at the predicted scale, are compared with the map point's representative descriptor *D*. Each map point may be seen in several keyframes depending on the viewpoint. Hence, it has more than one descriptor associated. To accelerate the matching, each map point holds a representative descriptor *D*, whose hamming distance is least with respect to all associated descriptors (Mur-Artal et al. 2014).
- The map point is associated with the best match.
The optimization of camera poses is performed by motion-only Bundle Adjustment (Appendix).

Covisibility graph

It represents the covisibility information existed between keyframes. It is an indirectly weighted graph, and it is essential for many tasks, like defining a local map and enforcing the place recognizer performance. It consists of nodes and edges. Each node is a keyframe and an edge between two nodes exists if the two keyframes observe the same map points (at least 15). The weight of each edge, θ is the number of shared map points. As θ descends, the graph becomes more interconnected. When a new keyframe is inserted in the system, it is linked to the already existed keyframe which shares the most map points with. Likewise, when a keyframe is erased, the system updates the links affected by that.



(a) (b)

Figure 18. (a) Reconstructed map: keyframes (blue), current processing keyframe (green), map points (black and red). (b) Covisibility graph: nodes and edges (green) (Mur-Artal et al. 2017).

5. Finally, the camera pose of current frame is optimized using all the above correspondences. The refinement minimizes the reprojection error retaining fixed the positions of map points.
6. A keyframe selection is performed. The system extracts as much as possible keyframes and then it discards all the keyframes that are redundant e.g., those which offer only few information about the scene recorded. More specifically, a frame is considered as a keyframe when all the following conditions are satisfied:
- After a relocalization takes place, more than twenty frames should have been processed.
- More than twenty frames must have passed from last keyframe insertion (Mur-Artal et al. 2015).
- At least 50 points must have been tracked in the current frame.
- Current frame tracks less than 90% points than K_{REF} (Mur-Artal et al. 2015).

In the case of stereo SLAM, a new keyframe is inserted when tracked close points are less than 100 and the current frame can create at least 70 new close stereo points. This is indispensable for handling environments where a big part of the scene is far from the stereo sensor (Mur-Artal et al., 2017). An adequate number of close points is very important, in order to accurately estimate the camera's translation.
In stereo and RGB-D cases, keypoints are classified into close and far. In general, a stereo keypoint is defined by three coordinates $x_S = (u_L, v_L, u_R)$, where u_L and v_L are the coordinates in the left image and u_R is the horizontal coordinate in the right image. In RGB-D cameras, u_R results from the transformation of image's depth d into a virtual right coordinate as follows:

$$u_R = u_L - f_x * \frac{b}{d}$$

, where f_x is the horizontal focal length and b, the baseline between the structured light projector and the infrared camera (Mur-Artal et al., 2017). Hence, the uncertainty of depth estimation is transferred into the virtual right coordinate. As close point is considered the point whose depth is less than 40 times the stereo or RGB-D baseline, otherwise it is considered as far. It is easily understood that far points deliver poor scale and translation information contrary to close points that are accurately estimated and provide accurate scale, translation, and rotation information. So, far points are only triangulated when they are seen by multiple views.
Local Mapping

The local mapping thread consists of processing new keyframes. Keyframes are added in the system by the tracking and stored in a queue. Local mapping starts processing the oldest in the queue keyframe and execute the following steps:
1.With a keyframe insertion, the covisibility graph is updated by adding a new node and the edges derived from the shared map points with other keyframes (Mur-Artal R. et al. 2015). The spanning tree is then updated by connecting the new keyframe with the one with most map points in common. The spanning tree is a sub-graph of the covisibility graph with minimal number of strong edges.

Figure 19. Spanning tree (Mur-Artal et al. 2017).

2. The new processing keyframe is converted into its bag-of-words representation. Now, the image contains a bag-of-words vector, as it is explained in DBoW2 review. This vector will be used later by loop closing procedure. Apart from that, it will boost data association for triangulation by linking each ORB feature point to the second node in the vocabulary tree.
3. A map point culling procedure is executed. Recent map points pass a test during the first three keyframes after creation. Map points, to be maintained, should satisfy the following two states:
- In tracking thread, the point must be found in more than 25% of the frames in which it is predicted to be detected.
- The point should be visible in at least three keyframes, if more than one keyframe have been processed after its creation.
The only case, where a map point that has passed the test can be deleted is, when it is observed from less than three keyframes. This can be arisen from either the keyframe culling procedure or Bundle Adjustment (when it removes the outliers).
4. The triangulation of ORB correspondences from connected keyframes in the covisibility graph creates new map points expanding the reconstructed map. The number of keyframes should be bounded because in complex maps (very interconnected) the computational cost would be excessively big i.e., the amount of connected keyframes would be huge. For each unmatched ORB feature in the current keyframe, a search with other free ORB points in other keyframes is operated just using bag-of-words recognition procedure i.e., the direct index (comparing ORB features that belong to the same node at the second level of the vocabulary tree). All the matches should satisfy the epipolar geometry otherwise they are discarded. Once ORB matches have been found, they are triangulated. To be accepted, map points must pass several checks, as the positive depth in both cameras, low reprojection error, sufficient parallax (more than one degree) and scale consistency. A map point can be seen in more than two keyframes. In order to find its projection in other keyframes, it is projected in them, following the 4^{th} step of tracking module.
5. A local Bundle Adjustment (Appendix) is performed, optimizing the current keyframe's pose, the pose of all keyframes connected to it in the covisibility graph and all the map points observed from these keyframes. Keyframes that are not linked with the current processing keyframe and see the same map points, are set fixed in the non-linear optimization. For optimization, Mur-Artal et al. (2014) use the Levenberg-Marquadt method applied in g2o and the Huber cost function.
6. To enhance Bundle Adjustment procedure (becomes more complex as the number of keyframes is growing) and allow a lifelong operation in the same environment (new keyframes

are added only when the visual content of a scene changes), unnecessary keyframes are discarded. To be more specific, all keyframes whose 90% of the map points have been observed in, at least, other three keyframes in the same or finer scale are discarded, ensuring that keyframes which create map point with most accuracy, are retained (Mur-Artal et al. 2017).

Map Points
Each created map point of the system stores:
- its 3D real world coordinates
- the normal vector *n* which is the mean vector of its viewing directions
- a representative descriptor D described above.
- d_{min} and d_{max} distances at which the map point can be observed according to the scale and distance at which it has been seen.

Keyframes
Each keyframe stores:
- its camera pose T_{iw} which is a rigid body transformation that can transform all points from the world to camera's coordinate system.
- the camera intrinsic parameters i.e., focal length, principal point
- all ORB features extracted in this frame, either associated to a map point or not. Their coordinates are undistorted if a distortion model is provided (Mur-Artal R. et al. 2015).

Loop closing

The loop closing procedure consists of detecting loops in the reconstructed map and correcting them, performing global optimization (global/full Bundle Adjustment). In this way, the global consistency of the map is achieved. After conducting some experiments on popular datasets, Mur-Artal et al. (2015) deduced that the existence of loop closures is indispensable for accurate reconstructions, e.g., drift correction. Loop closure starts processing the last processed keyframe by the local mapping and executes the following tasks:
1. First, a loop must be detected. To achieve this, a similarity transformation between the bag-of-words vector of current keyframe and all its adjacent keyframes in the covisibility graph, is computed. The lowest score, s_{min} is retained. The system then, queries the keyframe database by the recognition procedure for loop candidates which should be consistent with three previous keyframe matches (Mur-Artal et al., 2014). All keyframes whose score is lower than s_{min} and are directly connected to current keyframe in the graph, are abandoned. As in relocalization operation, there can be several loop candidates, if similar places have been recorded.
2. To successfully close a loop, a similarity transformation or geometrical validation (Mur-Artal et al., 2017) between the current and the candidate keyframes must be computed. In monocular SLAM, the map can drift in seven DoFs, i.e., three translations, three rotations and scale. At first, correspondences between ORB features associated with map points in the current keyframe and the loop candidates are found through the bag-of-words recognizer. At this moment, there is a set of 3D-to-3D correspondences for each loop keyframe candidate. A similarity transformation (Sim3) described below, is finally calculated using the method of Horn into a RANSAC scheme for each candidate.
Similarity Transformation:

$$S_{k.l} = \begin{bmatrix} s_{k.l} \times R_{k.l} & t_{k.l} \\ 0 & 1 \end{bmatrix}$$

, where k is the current keyframe, l is the loop candidate, $s_{k.l}$ is the scale factor, $R_{k.l}$ is a rotational matrix and $t_{k.l}$ is a translation vector (Mur-Artal et al., 2014). The 3D-to-3D correspondences are used for the transformation estimation. The Sim (3) with most inliers is accepted. A match is considered as an inlier if the sum of reprojection error in the current and candidate keyframe is less than 6 pixels. Also, if RANSAC computes a transformation supported by the 40% of the initial correspondences in less than 70 iterations (Mur-Artal et al., 2014), the former is

considered successful. Afterwards, it is optimized by minimizing the reprojection error in both keyframes as:

$$e_1 = x_{1,i} - \pi_1 * (S_{12}, X_{2,j}) \;\&\; e_2 = x_{2,i} - \pi_2 * (S_{12}^{-1}, X_{1,i})$$

and the cost function given n matches i(keypoint)→j (associated 3D map point), as:

$$C = \sum [\rho_h * (e_1^T * \Omega_{1,i}^{-1} * e_1) + \rho_h * (e_2^T * \Omega_{2,j}^{-1} * e_2)]$$

, where $S_{12}$is the relative transformation, $\Omega_{1,i}$, $\Omega_{2\,j}$ are the covariance matrices associated with the scale in which the keypoints in images 1 and 2 were detected. In this optimization, the points are fixed (Mur-Artal et al. 2017). Then, a guided search of more matches is performed, and the transformation is refined again.

In stereo and RGB-D SLAM where depth is known, the scale information is observable and so, there is not a scale drift. The geometric validation and pose-graph optimization count on rigid body transformations instead of similarity transformations.

3. After, the detected loop must be corrected. This consists of fusing duplicated map points, updating, and inserting new edges in the covisibility graph. Before fixing the loop, the loop closure informs the local mapping to stop after its current processes. This is very crucial because both threads would change keyframes' and map points' position disrupting the consistency of the map. The similarity transformation is applied to the pose T_{iw} of the current keyframe (K_i) and to all neighbors of K_i so that both sides get aligned. All map points observed by the loop candidate keyframe and its neighbors are projected into the current keyframe and its neighbors. A search for matches in a determined area around the projection is performed. All these matched map points and map points that were inliers in similarity transformation estimation are fused.

4. A global optimization must be performed to effectively close the loop and correct the accumulated error. Due to this drift, the already reconstructed map is far from the reality meaning that the use of a robust cost function for BA is not possible. To facilitate the cost function's convergence, an initial solution must be computed. This initial solution derives from the result of the pose graph optimization, based on the essential graph. For this, the ORB-SLAM 2 uses the g^{2}o. This distributes the error along the graph. The optimization is based on similarities transformations to fix the scale drift. In particular, the absolute transformation of all poses, T_{iw} is converted into their similarity transformation, $S_{i,w}$ retaining the translation and rotation and setting the scale to 1. Then, the relative transformation $\Delta S_{i,j}$ between two poses is computed. To correctly close the loop, the relative transformation (the difference of their similarity transformations) between the current and the loop keyframe is computed. The residual error $r_{i,j}$ between two poses $S_{i,w}$ and $S_{j,w}$ with respect to the constraint $\Delta S_{i,j}$ is:

$$r_{i,j} = log_{Sim(3)}(\Delta S_{i,j} * S_{j,w} * S_{i,w}^{-1})_{Sim(3)}^{v}$$

, where $log_{Sim(3)}$: $Sim(3) \rightarrow sim(3)$ maps from the over-parametrized representation of the transformation to the tangent space and $()_{Sim(3)}^{v}$: $sim(3) \rightarrow R^7$ is a vee-operator that maps from the tangent space to the minimal representation with the same elements as the DoF of the transformation (Mur-Artal et al., 2014). The residual error should be minimized likewise the cost function:

$$\chi^2 = \sum_{(i,j)} r_{i,j}^T * \Lambda_{i,j} * r_{i,j}$$

, where $\Lambda_{i,j}$ is the covariance matrix of the residual $r_{i,j}$ and is set to the identity matrix. Initially all the residuals are zero except from loop's residual. Then, all the poses are optimized to distribute this error along the graph (Mur-Artal et al. 2014). The loop keyframe pose is excluded from the optimization and so, this variable should be disappeared from the linear system. The Levenberg-Marquadt algorithm (Levenberg et al., 1944) implemented in g^{2}o (Kümmerle et al. 2011) is used for the optimization's solution considering the solver CHOLMOD (Cholesky decomposition) for the sparse structure of the problem. This sparsity derives from the sparse

covariance matrix and allows the efficient solution of the linear system. After the optimization of the poses, the 3D map points should be corrected. For each map point x_j, the corrected map point x_i^{cor}, selecting a keyframe in which the point has been seen, is computed as follows:

$$x_i^{cor} = (S_{i,w}^{cor})^{-1} * T_{iw} * x_j$$

,where $S_{i,w}^{cor}$ is the optimized similarity transformation of the keyframe and T_{iw} is its pose. Finally, the corrected similarity transformation is converted into the 3D rigid body transformation T_{iw}^{cor} as follows:

$$S_{i,w}^{cor} = \begin{bmatrix} s*R & t \\ 0 & 1 \end{bmatrix} \rightarrow T_{iw}^{cor} = \begin{bmatrix} R & \frac{1}{s}*t \\ 0 & 1 \end{bmatrix}$$

In stereo and RGB-D cases, where scale factor is known, the pose graph optimization is based on rigid body transformations only instead of similarity transformations.

Essential graph

In cases that covisibility graph is very dense, meaning high computational cost for many tasks, the essential graph is required. It is derived from the covisibility graph and contains a subset of edges while maintaining all the nodes (spanning tree). It uses edges that represent high covisibility between the connected keyframes, i.e., θ_{min}=100. Also, it includes the loop closing edges.

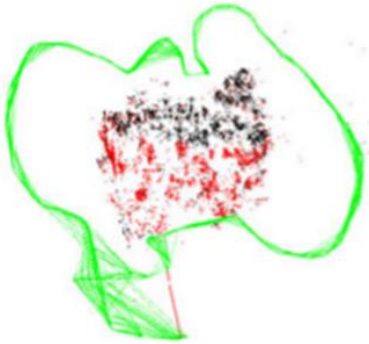


Figure 20. Essential graph: nodes and edges (green), loop closing edge (red), map points (black and red) (Mur-Artal et al. 2017).

5. A global Bundle Adjustment is executed in a separated thread due to its high computational cost, using the above formulation for optimizing poses and points. This allows the simultaneous performance of mapping and detecting loops. So, it is necessary to merge the Bundle Adjustment's solution with the current state of the map (Mur-Artal et al., 2017). If a loop is detected while the optimization is running, the system terminates the optimization procedure and proceeds to loop's correction which will release the full BA again. After the termination of optimization, the updated keyframes and map points i.e., optimized by BA, should be merged with those which are non-updated and were inserted during the optimization. This is achieved by propagating the correction of updated keyframes to non-updated points and keyframes through the spanning tree (Mur-Artal et al., 2017). In Mur-Artal et al. (2017) study, some conducted experiments have shown that the solution of the pose-graph optimization is so accurate that the full BA optimization shortly improves the solution.

Description of DBoW2 place recognizer-Thirdparty:

In many tasks of ORB-SLAM 2 e.g., relocalization and loop closing procedure, a place recognizer is required, so that revisited places can be detected as soon as possible. For this reason, the system makes use of DBoW2 algorithm, presented by Galvez-López et al. (2012) using ORB features, which is a place recognizer method. Specifically, the processed image is converted into a bag-of-words representation i.e., a vector of visual words. This method relies

on a vocabulary tree that is constructed off-line by converting a binary descriptor space into visual words resulting in a more flexible vocabulary. The vocabulary tree consists of several levels, nodes and leaves. The vocabulary is structured using feature points from several training image datasets which are different from those which will be processed later. The first level of nodes results from the classification of features' descriptors into k clusters using K-means algorithm. The other levels of the vocabulary tree derive from the classification of descriptors of each node. The number of iterations determine the number of levels. The "nodes" in the last created level constitute the visual words. To be aware of the importance of a word, a weight must be given. Precisely, the weight of a word is being decreased when the latter is appeared very often in images. Hence, for each word, a term frequency-inverse document frequency (tf-idf) is formed, where tf stands for the frequency in the current image and idf, for the frequency in the whole image dataset. So, the score is higher when the word is very frequent in the image and very rare in the dataset. Also, the system forms the direct index for each image that stores the nodes at level l that are ancestors of its visual words and its features associated with each node (Galvez-López et al., 2012). However, the first step is to convert the image into its visual words. For this, its binary descriptors "run" the vocabulary tree and choose for each level the nodes that minimize the Hamming distance. The system creates incrementally a database of keyframes that is always updated e.g., when unnecessary keyframes are removed during local mapping or when a new keyframe is inserted; and it holds an inverse index for each word, that stores in which keyframes the word has been seen. This facilitates the database querying procedure reducing the search time. To detect a revisited place, a similarity between the bag-of-words vectors of a current keyframe (v_i) and others in the database (v_j) is computed as follows:

$$s(v_i, v_j) = 1 - \frac{1}{2} * \left| \frac{vi}{|vi|} - \frac{vj}{|vj|} \right|$$

This score is then normalized with the best score that is supposed to be obtained from an image that shows the exact same place i.e., the previous processed image assuming that an image sequence with overlap is processed. The normalized similarity score is:

$$\eta\,(v_i, v_j) = \frac{s\,(v_i, v_j)}{s\,(v_i, v_{i-1})}$$

So, sequential images may have similar scores. DBoW2, based on this, groups images that are close in time, the "islands" (Galvez-López et al., 2012) and calculates only one score for the group which is the sum of their individual scores. Eventually and after querying the database, the highest score is depicted and the image with the highest individual score is regarded as a loop candidate. Though, individual scores must be higher than a threshold value otherwise they are rejected. Also, it should be mentioned that the $s\,(v_i, v_{i-1})$ score must achieve a minimum value for images to be considered, otherwise they are skipped. This is done because a very small $s\,(v_i, v_{i-1})$ score can wrongly lead to very high $\eta\,(v_i, v_j)$ scores. The recognition procedure returns all keyframe candidates whose scores are higher than the 75% of the best score. However, the best group has to pass a temporal consistency test, meaning that it has to be consistent with k previous queries i.e., best groups should create an overlapping sequence. Finally, a geometrical verification is performed to accept a loop candidate. This is achieved by computing a fundamental matrix inside a RANSAC scheme with at least 12 correspondences. These correspondences can be efficiently found with the use of the direct index comparing only those features that belong to the same node in the level l. The execution time and the number of found matches depend on the determination of the level of the vocabulary tree.

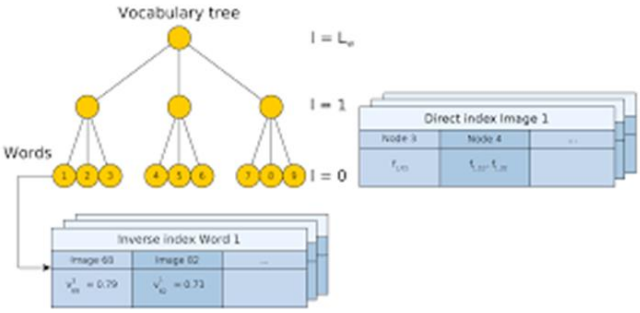


Figure 21. Structure of the vocabulary tree of DBoW2, which consists of levels, nodes and leaves (visual words) (Galvez-López et al. 2012)

Description of ORB:

ORB – an extremely fast binary descriptor – is first presented by Rublee et al. (2011) and is based on the well-known FAST detector and the BRIEF descriptor, with some modifications, to be rotation invariant and tolerant to noise.

FAST detector was originally developed by Rosten et al. (2006) and its promising advantage is its computational efficiency (i.e., high-speed performance). Thus, it is suitable for real-time applications. It uses one single parameter, the intensity threshold between the central pixel and its neighbors in a circle (radius of 9) around it. More precisely, a pixel is selected to be the interest point and its intensity I_p is measured. A threshold value (t) is then defined. If the intensity of N neighbor pixels in the circle is brighter than I_p+t or darker than I_p-t, then the pixel is considered as a corner. In order to increase the performance's speed, the intensity of the neighbors pixels is calculated in a specific way. This procedure is performed for all the pixels. So, it is concluded that FAST produces a lot of edges but not confident corners. This happens because it requires a circle of contrasting pixels more than three quarters around the center of corner. Hence, Rublee et al. (2011) implemented the Harris corner algorithm to order the FAST keypoints. Similar to ORB-SLAM 2 implementation, a scale-pyramid is employed, in order to produce multi-scale features. They built on FAST by giving an orientation of a corner using the intensity centroid measurement which assumes that the corner's intensity is not at its center (i.e., is offset). The vector linking the corner and the centroid is exploited for determining the orientation. Firstly, the moments of a patch are defined as:

$$m_{pq} = \sum_{(x,y)} x^p * y^p * \mathrm{I}\ (\mathrm{x}, \mathrm{y})$$

, which are then used for the centroid: C = [$\frac{m_{10}}{m_{00}}$, $\frac{m_{01}}{m_{00}}$]. Therefore, the orientation of a patch is

$$\theta = \mathrm{atan2}(m_{01}, m_{10})$$

Also, x and y should be retained inside the circle.

BRIEF descriptor converts a patch into a binary vector (a string of bits). The patch is defined as a square of pixels around the keypoint. Once an image has been smoothed, the vector is built up using the binary test results, which are of the following form:

$$\tau(p: x, y) = \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x) \geq p(y) \end{cases}$$

, where (x, y) is a random pair of pixels inside the patch and p(x) is the intensity of the patch at a point x. The vector is described by:

$$f_n(p) = \sum_{(1<i<n)} 2^{i-1} * \tau * (\mathrm{p}: x_i, y_i)$$

The Gaussian distribution is selected for the tests. To achieve descriptors invariant to rotation, Rublee et al. (2011) proposed a learning method for features, called rBRIEF, so that the selected

binary tests being less correlated and with higher variance (i.e., more discriminative point). They set up a training set extracting 300.000 keypoints from several images and calculated all possible binary tests in a patch of 31×31 size discarding those that were overlapped. The process of the algorithm is the following:
- each test is performed for all the training patches
- the vector T is formed by ordering the test with respect to their distance from a mean of 0.5
- the first test is selected and included in the final vector R (and removed from the T), the next test is compared with all the other tests in R and is preserved if its absolute correlation is lower than a threshold value; and this is done until the vector R is formed with 256 tests. Otherwise, the threshold value becomes higher.

4. Development and implementation of the ORB-SLAM 2 system and Visual Map

4.1. System Development

4.1.1. Map Save Function

During the implementation of the system, a viewer opens showing the 3D point cloud (map) creation, as well as the keyframe trajectory. Unfortunately, when its process ends, the generated map is deleted immediately. However, a file that contains the trajectory is created. Therefore, when the SLAM process ends, a function that allows the map saving in a PCD file (Point Cloud Data, 2021) was built. This function was integrated into the "system" class of ORB-SLAM 2, which handles all the main functionalities; in such way that it can efficiently save the map. The function uses the ORB-SLAM class MapPoint, from which the GetAllMapPoints variable is leveraged.

The PCD file is a file format that supports 3D point cloud data. Its supported version by the PCL library is 0.7. It contains several headers that declare certain properties of the point cloud data that were stored in the file. These headers should be encoded into ASCII format. Each header is at a different line inside the file. More specifically, the generated file contains the following headers:

- its **version**, i.e., 0.7
- the **fields** that identify the name of each dimension that a point cloud has. In particular, the fields presented in this file are x, y and z coordinates of map points.
- the **size** that specifies the dimension's size in bytes, i.e., 4 since the map points are declared as unsigned float.
- the **type** that declares the types of each dimension as a char. The type is set to F as map points are float numbers.
- the **count** that stands for the number of components, a dimension has. By default, it is set to 1 since it does not exist.
- the **width** and **height**, that are used together. In the current file, the width specifies the total number of map points of the point cloud, as the latter is unorganized. In case that the dataset is organized, the width presents the total number of points in a row. In general, an organized point cloud is data that are split into rows and columns, e.g., data generated from stereo cameras. In general, the height defines the total number of rows for organized point clouds whereas, in this file is set to 1.
- the **viewpoint**, that determines an acquisition viewpoint for the map points, i.e., 0,0,0,1,0,0,0.
- the **points**, that declare the total number of points in the point cloud.
- the **data**, that specifies the data type of the point cloud. In the current file, the data are stored in an ASCII format.

It is important that the headers be entered in the above specified order. After the last header, the 3D coordinates of map points are presented. An example of a sequence of the TUM dataset is presented below. The PCD file can be viewed in several softwares that handle point cloud data, such as the CloudCompare (CloudCompare, 2021) or using PCL viewer through the command line or through a text editor.

```
VERSION .7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 1573
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 1573
DATA ascii
0.1195603 -0.02861849 1.007299
0.2608371 -0.05850596 0.946375
0.4075675 -0.01130074 0.9759435
-0.453173 -0.05310699 0.9536226
0.4112271 -1.090627 3.077947
0.3259748 -1.402775 4.024253
-0.4046929 0.221074 0.7554911
0.3066454 -0.06163505 0.9384912
```

Figure 22. Part of the PCD file of TUM dataset.

4.1.2. Incremental visual map using ORB-SLAM 2

The feature-based SLAM algorithms rely on the extraction of feature points and thus, they do not exploit the whole information of an image. The ORB-SLAM 2 is such an algorithm which is based on a subset of the input frames, called keyframes, in order to reduce the computational cost and make the procedure work in real-time. Therefore, it produces a sparse point cloud from which one cannot extract critical information about the examined environment. In order to understand the scene and its characteristics better, one should combine the generated point cloud with the images captured from a visual sensor or, have his attention during the whole execution of the system so to remember the area that has been reconstructed up to that time. This is very inconvenient, especially for large regions. For this reason, this diploma thesis proposes the creation of an incremental visual map based on the ORB-SLAM 2 system. The visual map helps to visualize mapped areas, offering a more simple and informative view of the environment to the user and allowing further analysis. It is basically the alignment of multiple images to generate a larger image of the scene and can be obtained by realizing the geometric relationships between images. The following figure illustrates the main steps of the proposed approach.

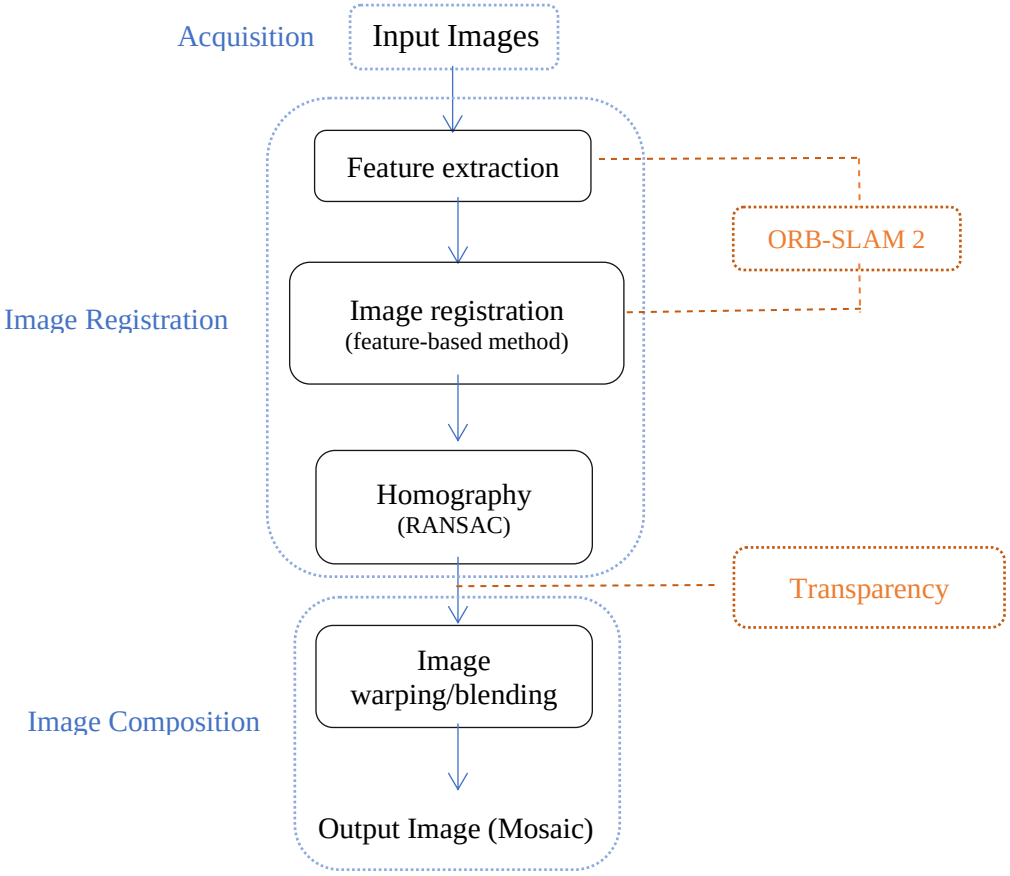


Figure 23. Flowchart of the proposed approach.

The feature extraction is the first step involved in image composition. It consists of extracting distinct features from the images and can be used for estimating the relative transformation between them. Its main benefit is that it describes a given data set with the minimal information. The data set can be described by corners, edges or blobs regarding the required accuracy of the application and the execution time. There are several detector algorithms that can efficiently perform the feature extraction procedure and are used depending on the type or the problem of the application. According to (Kota et al., 2016), the desired property of such a detector is the repeatability, i.e., whether the same feature point will be obtained from images that depict the

same area. Since the aim of this diploma thesis is to connect the visual map with the ORB-SLAM 2 algorithm, the correspondences obtained from its process are exploited.

The image registration is the main task of the procedure. It is responsible for establishing geometric correspondences between images that show the same scene. This is essential for images so to be transformed and analyzed into a common reference plane/frame, which can be one of the processing images. The most common transformation for registering the images is the homography.

The homography estimation is crucial, to transform an image plane into another. It is a 3×3 matrix with 8 unknown parameters that describes the translation, orientation and scale. The application of the homography to the pixels of an image results in the creation of a new image transformed, according to the reference plane. It describes only planar or near planar scenes. In reality, it is very rare for an area to be exactly planar. Therefore, an assumption is made, to get the best possible results. The relation that links the corresponding points between two images is:

$$p' = Hp, \text{ where } H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

For the estimation of the 8 parameters, at least 4 correspondences should be obtained. If more correspondences are available, a linear least square problem is solved. In reality, most of the time the correspondences contain outliers that should be discarded, to preserve an erroneous homography estimation. For this reason, the transformation is estimated inside a RANSAC scheme, in order to detect the spurious matches.

The final step is the image warping and blending processes which lead to the final image generation. It consists of defining the dimensions of the final output image which can be obtained by computing the range of the transformed coordinates of each input image. The difference of the minimum and maximum x and y values are the coordinates of the new image. The last step is to "paint" and visualize the final composed image.

As mentioned before, after the completion of the ORB-SLAM 2 algorithm two files are produced. The first one contains the position and orientation of the keyframes (i.e., extrinsic orientation) while the second one provides the 3D coordinates of the map points (i.e., the sparse point cloud). The aim of the proposed approach is to build a visual map using only these keyframes produced by the examined algorithm, instead of using all the input frames. The keyframes that survive during the whole procedure, go through two very important processes. The first is the keyframe selection which is performed during the tracking thread. According to Mur-Artal et al. (2017) a frame is considered as keyframe when 1) more than 20 frames have passed from the last keyframe insertion, 2) more than 20 frames have been processed after a relocalization, 3) at least 50 points have been tracked in this and 4) it tracks less than 90% of points than reference keyframe. These conditions ensure a good relocalization (1), a robust tracking (3) and mapping (2) and a minimum visual change (4). The system tries to insert keyframes as soon as possible, to make the tracking procedure more robust and resilient to challenging camera movements, such as pure rotations. The second check that the keyframes pass is a culling procedure, which is responsible for a restricted map reconstruction. More specifically, the ORB-SLAM 2 rejects all the keyframes whose 90% of map points have already be seen in at least other three keyframes in the same or finer scale. In other words, it adds keyframes only when the visual content of the scene changes. These two procedures boost the performance of the Bundle Adjustment by reducing its computational cost. Especially in cases where an operation in the same environment is required, they will "protect" the 3D map to grow unbounded. For this reason, although the visual map is reconstructed using only the keyframes, it adequately covers the whole examined area.

Proposed Approach

Since the long-term goal of this thesis was to integrate the visual map into the ORB-SLAM 2 system so that the user understands if he has been through the same area again, in real-time, an approach was proposed, so as to leverage the feature matching procedure of the SLAM system. The main steps are the following:

1. **Read input keyframes**. The file of the keyframe trajectory produced after the completion of the ORB-SLAM' s 2 execution, contains the timestamp of each keyframe (i.e., its name), its position and orientation in a quaternion form. Therefore, the proposed algorithm reads the keyframe names from this file and saves them in a list.

2. **Image registration.** The geometric correspondences between two successive keyframes are established. For this, the matches from the ORB-SLAM 2 are exploited. The ORB-SLAM 2 system uses 7 different functions for feature matching whose input data change depending on the procedure being performed. The functions are:

(a) *SearchByProjection*. This function searches for matches between the current processing frame and the keyframes. It projects the local map points into the current frame and selects a match near the projection according to the descriptor distance, increasing the map points of the current frame.

(b) *SearchByBoW*. It matches the feature points of the keyframe and the current frame through the bag-of-word representation of the images, using the method described in Section 3.

(c) *SearchForInitialization*. It finds the corresponding feature points between two frames during the initialization procedure. It finds matches between two initial frames by calculating the Hamming distance between their descriptor vectors.

(d) *SeachForTriangulation*. It finds matches using the bag-of-word method between the ORB features of the current keyframe which have not map points associated and the unmatched ORB features of the previous keyframe. It is used by the Local Mapping procedure in order to expand the local map. In order for the map points to survive, they have to pass certain checks such as, the positive depth in both cameras, low reprojection error, sufficient parallax and scale consistency. This ensures that the matches are good, decreasing the number of outliers.

(e) *Fuse*. During the loop closing procedure where duplicated map points occur, this function project the map points seen by the candidate keyframe and its neighbors in the covisibility graph into the current keyframe and its neighbors. If the map point corresponds to a keypoint which has not a map point associated, the map point is created. If it corresponds to a keypoint which has a map point, then the map points are fused (Section 3-Loop Closing).

(f) *SearchBySim(3).* During the loop closure, some matches are not detected by *SearchByBoW* therefore, this function finds the area in the first keyframe where the feature points of the second keyframe will be and vice versa.

Since the visual map will be created using a subset of the frames, i.e., keyframes, the *SeachForTriangulation* function was exploited to get the matches. All the other functions do not accept as input two consecutive keyframes except for *SearchBySim(3)* which is performed only when loops are detected. To propose a generalized approach, this function was not selected.

- For this reason, a function that saves these matches into a text file, called "homography.txt" was integrated into the Local Mapping source code and specifically, into the

CreateNewMapPoints function. The text file is generated during the execution of the ORB-SLAM 2. The format of the file is the following:

keyframe1 keyframe2 (x1, y1) (x2, y2)
.
keyframe2 keyframe1 (x2, y2) (x1, y1)

keyframe2 keyframe3 (x2, y2) (x3, y3)
keyframe2 keyframe3 (x2, y2) (x3, y3)
.
.
.

- Then, another function, called *findtracks* was created. It takes as input the "homography.txt" file and the keyframe trajectory file and stores each pair of consecutive keyframes based on the latter file (i.e., keyframe1 & keyframe2, keyframe2 & keyframe3 etc.) with their correspondences, regardless if the keyframe is first or second stored in the "homography" file. It also searches and marks the pairs of keyframes for which either matches have not been found or are less than 4 (minimum required), with the word "*gap*". This is caused by the keyframe culling procedure of the ORB-SLAM 2, which is performed after the triangulation of new points. Hence, some keyframes in the "homography.txt" file will not be contained in the final keyframe trajectory file. For this reason, certain pairs of keyframes of the keyframe trajectory file do not have correspondences in the "homography.txt" file.

- The function *fillthegaps* reads the above list and finds matches for the gaps by performing the ORB operator with FLANN method. It extracts keypoints between the two keyframes and obtains their correspondences (number?) using the descriptor vectors. The "finaltracks.txt" file is generated with all the keyframe pairs and their correspondences.

- Finally, a function, called *loadtracks*, that loads the keyframe pairs with all their matches, was created so that can be used in the calculation of the homography.

To be ensure that the matches generated from both the ORB-SLAM 2 system (bag-of-word) and the ORB operator are correct, a function called *checkmatches* was created for plotting them on the corresponding keyframe pairs. This function takes as input the "finaltracks.txt" and the images (i.e., keyframes) and it outputs the image pairs on which the matches with a red circle are drawn.

3. **Homography estimation**. A function called *getHomo* was integrated and computes the homographies between two consecutive keyframes given the set of corresponding points, as they result from the above process. Therefore, all the homographies between keyframe pairs from the "finaltracks.txt" are calculated using the OpenCV library, inside a RANSAC scheme to reject the outliers (i.e., H21, H32, H43, etc.).

4. **Image warping/blending**. The first step is to define the reference image/plane on which all the keyframes will be projected. Here, the first image is selected. A function called *getRefHomo* was created. It takes as input the above estimated homographies and computes the homographies of all images with respect to the first one (referenced), i.e., H31, H41, H51 etc. as follows:

$$H31 = H32 \cdot H21$$

$$H41 = H43 \cdot H31$$

.
.

Once the homographies have been computed, the corners of each image are transformed to the first image plane using the corresponding homography (*calcBoundary* function). That is, the

boundaries of the second image are multiplied by the H21, the borders of the third image, by H31 etc. Their coordinates are first become homogeneous so that they can be multiplied by the 3×3 matrix.

Based on the transformed corners of each image, the proposed approach estimates the minimum and maximum x and y pixel values (*getBoundaries*). Then, it computes the dimensions of the final image, whose width will be the difference between the maximum and minimum value of x and height will be the difference between the maximum and minimum value of y (Kota et al. 2016). Afterwards, the function called *getWarpedImgs* takes as input the frames, the transformed homographies with respect to the first image and the calculated dimensions. Before projecting the whole image into the first image plane, it multiplies the homographies (referenced) with some offset. The homography maps part of the image to negative x, y values which are outside the image area and thus, they cannot be plotted and visualized. For this, an offset by some pixels is applied to the estimated homographies, in order to shift the entire warped image into positive coordinates (i.e., inside the image area) (Amiri et al. 201. The thing that should be done is to "construct" the homography matrix for a translation by this offset and multiply it by the original homography matrix. The homography with the offset has the following format:

$$offset = \begin{bmatrix} 1 & 0 & offset_x \\ 0 & 1 & offset_y \\ 0 & 0 & 1 \end{bmatrix}$$

The multiplication is performed as follows: *offset*refH,* where *refH* are the homographies with respect to the first image.
Now, it projects the whole images into the plane of the first image by applying to them, the homographies and using the double of the calculated dimensions, so that none of the images get cropped, i.e., fit in the overall image produced. In order to visualize the first image, it is multiplied by the identity matrix.
In order to paint/produce the composed image, the function *paintVim*:
- reads the pixels of the final image one by one.
- assumes that the color is black and searches in which of the warped images the pixel has color in the corresponding position. More precisely, for the (0, 0) of the final image, it reads the (0, 0) pixel of the first image and if there is a color value associated, it paints the pixel and stops the searching. Otherwise, it continuous searching in the first pixel of the second image etc., until it finds a color value. This was computational heavy and therefore; another approach was developed to accelerate the process. This is the following:

Improvement 1:

In order to make the process incremental, for each warped image a function called *getColoredBoundaries* finds a mask, which indicates where and where not color exists. It uses the colored pixels for coloring the corresponding pixels of the final image (since the ratio warped image/final image is one by one). After the completion of the procedure, the visual map is displayed (i.e., it can be considered as pseudo-incremental).

Improvement 2:

To make the process faster and incremental, for each warped image, a mask which shows where and where not color exists is found. Then, a function calculates the boundaries of the colored image and makes it transparent. For this, a function called *getTranspImg* was developed. It creates a fourth channel (i.e., channel A) by detecting the total black pixels (i.e., 0, 0, 0) and makes them transparent Based on these boundaries, the final image gets colored according to the respective warped image. Therefore, the final image will not include black parts (between the images). This procedure is done for every keyframe pair and then it is displayed.

However, while the incremental process is well displayed, the final image cannot be saved properly. The information about transparency could not be rendered and thus, the output image includes some black parts. The reason behind this needs further investigation.
Despite that, the final "mosaic" can be saved from the above two approaches either normally or transparent.

4.2. Implementation of the ORB-SLAM 2 system

For the successful implementation of ORB-SLAM 2, some requirements were necessary, i.e., libraries to be installed and built such as, Pangolin (Pangolin 2021), OpenCV (OpenCV, 2021), Eigen (Eigen, 2021) and two thirdparties, the DBoW2 and g2o (Kümmerle et al. 2011). More specifically, Pangolin is a lightweight development library for managing videos, offering a progressive and simple 3D navigation handler. In ORB-SLAM 2 it is used for real-time visualization of graphical data and user interface. The library of OpenCV allows handling images along with feature points, providing a variety of tools and functions such as, image processing, camera calibration, detecting and matching salient features, Deep Neural Network creation etc. The thirdparty libraries were already included into ORB-SLAM 2 and therefore, no further built was required. They were modified and used for place recognition and non-linear optimization, respectively. Finally, Eigen, required by g2o, is a high-level library that enables mathematical operations like algebra, vectors, transformations etc., required for ORB-SLAM's optimizations. Subsequently, ORB-SLAM 2 was built on an Intel Core i5-5200U Laptop with 8 GB RAM providing executables for three popular datasets, i.e., TUM, EuRoC and KITTI dataset, for monocular, RGB-D and stereo mode. However, this diploma thesis concentrates on the monocular implementation. The UBUNTU operation system was chosen.

4.2.1 Implementation of the monocular ORB-SLAM 2 to the TUM and EuRoC datasets

For the system to be efficiently implemented in a dataset, the ORB vocabulary, the timestamps and a setting file should be provided. The ORB vocabulary is created by converting a binary descriptor space into visual words, as mentioned in the previous section. The setting file is necessary for passing all arguments to the algorithm. It includes several parameters needed for the system's performance which are the camera calibration and distortion parameters, other optional camera parameters like projection matrix and rectification matrix, the frames per second of the camera, the number of features extracted per image, the scale factor between levels in the scale pyramid, the number of these levels, the number of features depicted in each cell of the grid and some viewer parameters such as, the keyframe's size, the keyframe's line width, the graph's line width, the size of the created map points, the current camera's size, its line width and the viewpoints X, Y, Z and F. Some of them can be changed depending on the respective application (e.g., the number of extracted feature points, when the keyframes cannot be initialized due to small number of features). In case that the input images are already rectified, the distortion parameters are set to zero. A first approach was performed with the implementation of two popular datasets, that had also been tested by Mur-Artal et. al (2017) for evaluation, to check that the algorithm can efficiently operate.

TUM Dataset

The first dataset used in the implementation of ORB-SLAM 2 was the freiburg1_xyz sequence of the TUM dataset (Sturm et al. 2012) generated in TUM University of Munich, which was recorded for the evaluation of RGB-D SLAM systems. However, it can be efficiently incorporated in monocular systems, too. The images showing an office environment (three equipped desks), were recorded using a handheld Microsoft Kinect camera. The Kinect camera is composed of a near-infrared laser and an infrared camera including a color one between them. More precisely, the near-infrared laser projects a pattern to the recorded scene and the infrared camera records it. This pattern is known, so the disparity can be easily computed. The dataset

contains both RGB and depth images in resolution of 640×480 at a frame rate of 30 Hz, stored as 8-bit and 16-bit, respectively. It consists of 798 images. It also includes a ground truth trajectory, generated from a camera system with eight high speed tracking cameras. The cameras are calibrated. The color and depth images got aligned as they were observed from two different cameras. Its duration is 30.09 sec while the average translation velocity is 0.24 m/sec and angular velocity is 8.92 deg/sec. This sequence mainly contains translation motions while the orientation remained fixed. The ORB-SLAM 2 system was implemented to the dataset downloaded. A sample of the dataset images and the parameters in the setting file are presented in Figure 23. The system can operate at the same frame rate (30 fps) as the video was recorded.

Figure 24. Images from office sequence of TUM dataset.

In order to save the reconstructed map, the existing c++ file was modified by adding the map save function. According to the dataset, the calibration parameters and the fps in the configuration file were as follows:
focal length → f_x=517.306408, f_y=516.469215 pixels
principle point → c_x=318.643040, c_y=255.313989 pixels
coefficients of radial and tangential distortions →k_1=0.262383, k_2=-0.953104, p_1=-0.005358, p_2=0.002628, k_3=1.163314
fps → 30

Figure 25. Setting file of freiburg1_xyz sequence of TUM dataset.

Immediately after the implementation, a viewer that displays the camera poses in real-time and the 3D reconstructed map, opened along with a window. Specifically, the former, called map viewer showed the current camera's position in a green square, the previously estimated keyframe's pose in blue, the edges of the covisibility graph in green and the map points in red and black. Points colored red represented the reference reconstructed map points, that correspond to a local map. Tracking "uses" only these points, ignoring all the others. The rest were colored black. This window also gives the opportunity to the user to choose what one would like to see, i.e., the map points, the keyframes, the graph, to follow the camera and to activate or deactivate the localization mode. The operation of the localization mode disables

the local mapping and loop closing and allows only the tracking thread to track the camera in the previously mapped/reconstructed scene. The second window displayed every processing frame with its extracted feature points. Additionally, it showed the number of processed keyframes, the number of map points and the correspondences that had been found. Both are presented below.

(a) (b)

Figure 26. Viewers during TUM dataset implementation: (a) Map viewer of ORB-SLAM 2. (b) Current processing keyframe window.

After the completion of the system's performance, both windows closed directly, and two files were generated. The first one was a text file that contains the keyframes' name with its pose (i.e., the keyframe trajectory). The pose is defined by three components for the translation and four for the rotation. The orientation is referred to a quaternion. The second file was a PCD file that contains the 3D coordinates of map points, referred to the first keyframe's pose. ORB-SLAM 2 is initialized in an arbitrary coordinate system and due to the lack of external measurements, the generated point cloud is not represented in the true scale. Both the keyframe trajectory and map points were plotted into CloudCompare (CloudCompare, 2021), a 3D point cloud processing software which offers a set of several tools for visualizing, editing, and comparing point clouds and triangular meshes.

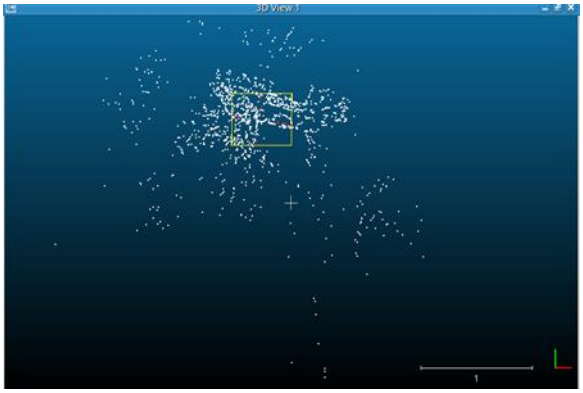


Figure 27. TUM's sparse point cloud (white) and its keyframe trajectory (red).

It can be seen that the 3D reconstructed point cloud was so sparse with no color information, that the comprehension of the mapped scene is unfeasible. However, according to the experiments and the results of Mur-Artal et. al (2015), it can be deduced that the system achieves high accuracy due to loop closure detection, i.e., detecting whether the sensor has returned to a previously visited location; to the pose-graph optimization, where the sensor's measurements are abandoned and the global Bundle Adjustment, where both camera poses and map points are optimized. Its absolute keyframe trajectory RMSE is 0.90 cm.

EuRoC dataset:

The second dataset implemented was a sequence (Machine Hall 1) of the EuRoC dataset (Burri et al. 2016) generated in an industrial environment of ETH University of Zürich and collected on board a Micro Aerial Vehicle (MAV). It was created for evaluating the performance of visual-inertial SLAM systems using MAVs, but also, it competently works in monocular cases since sufficient motion was carried out in the beginning of the recording. The dataset contains greyscale stereo images (11 cm stereo baseline), IMU measurements and a ground truth trajectory from a Leica Nova MS50 laser tracker which measured the position of a prism located on top of the MAV (with millimeter accuracy). The images (3682 in total) were captured by two global shutter monochrome calibrated cameras at a frame rate of 20 Hz and timestamped (nanoseconds) on board the MAV. Its duration is 182 sec while the average translation velocity is 0.44 m/s and angular velocity is 0.22 rad/sec. The scene recorded is described with brightness and informative texture. The system can operate at the same frame rate (20 fps) that the video was recorded.

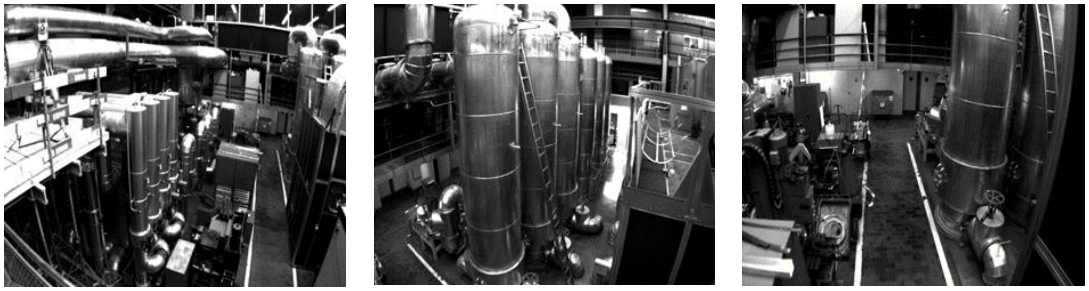


Figure 28. Images from one sensor of Machine Hall 1 of EuRoC dataset.

According to the dataset, the calibration parameters and the fps in the setting file were added as follows:
focal length → f_x=458.654, f_y=457.296 pixels
principle point → c_x=367.215, c_y=248.375 pixels
coefficients of radial and tangential distortions →k_1=-0.28340811, k_2=0.07395907, p_1=0.00019359, p_2=1.76187114e-05
fps → 20

The implementation of this sequence was performed in the same way as described before and both the keyframe trajectory and map points were plotted.

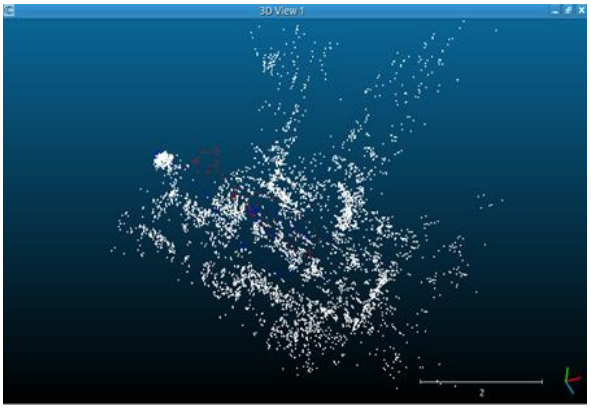


Figure 29. Machine Hall's sparse point cloud (white) and its keyframe trajectory (red/blue).

After the evaluation of ORB-SLAM 2 on three popular datasets, including the TUM and EuRoC datasets, and comparisons with other state-of-the-art monocular visual SLAM systems, Mur-Artal et al. (2015) demonstrated that their system achieves remarkably high accuracy for many different scenarios in comparison with other already existing systems. The accuracy of their

system is below 1 cm for small indoors environments and a few meters for large outdoors ones, assuming that the scale was first aligned with the groundtruth.

4.2.2 Implementation of ORB-SLAM 2 on underwater and aerial environment

In recent years, UAVs and ROVs equipped with visual sensors have received great attention from the community of robotics and computer vision due to their numerous applications. One essential feature of autonomous driving is the ability to operate safely in unknown environments and therefore, the localization accuracy is of great importance. Current works mostly use a Global Navigation Satellite System (GNNS) in order to directly estimate their absolute position. However, there are many cases, where the GNNS cannot deliver high accuracy or cannot work at all, such as urban canyons and underwater scenarios, known as GNNS denied environments. This problem can be efficiently addressed by Visual SLAM techniques. Hence, the examined SLAM system, ORB-SLAM 2 was implemented on two datasets that capture two different scenarios: an underwater and an urban.

A. AQUALOC Dataset

Overview of the dataset:
Another dataset used for the implementation of the ORB-SLAM 2 system and the visual map, was an underwater sequence of the AQUALOC Dataset developed by Ferrera et al. (2018). The raw images (11377 in total) in resolution of 968×608 show an archaeological site of a pile of amphorae, whose top is a few meters above the seabed in the Mediterranean Sea, off the shore of Corsica. It is located at a depth of approximately 380 meters, meaning that the environment is dynamic with both high-texture (amphorae) and textureless areas containing sandy regions around the hill with many fishes getting in the vicinity of the camera. The data acquisition was performed using a ROV (Remotely Operated Vehicles) in which a monochromatic camera with a fish-eye lens was integrated along with a pressure sensor (delivers depth data at 60 Hz), a low-cost IMU and an embedded computer, to record synchronously the sensor's measurements. The camera was placed behind a dome to minimize the distortion effects produced by the different refractive index of water and air. The image acquisition rate was 20 Hz while the IMU data were recorded at 200 Hz. The dataset also provides ground-truth data acquired, using the SfM technique offline. Its duration is 569 seconds, and its length is 122.1 meters. Moreover, it offers the timestamps of each image in nanoseconds, that were necessary for the implementation.

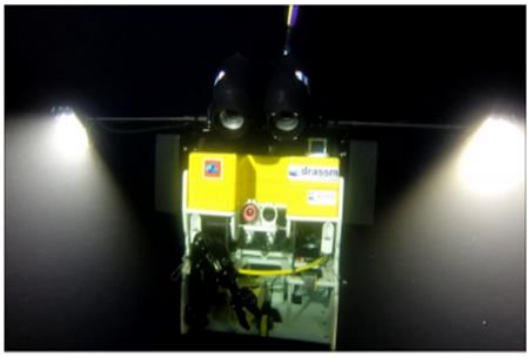


Figure 30. The Remote Operated Vehicle Perseo (Ferrera et al. 2019).

Description of the archaeological site:

The archaeological site is known as Capo Sagro 3 and is one of the few ancient deep-sea wrecks so far that are undamaged. The pile of Greek-Roman amphorae is 16-meter long and 9-meter wide. The amphorae were made for transporting wine along the Tyrrhenian coast of Italy around

180-170 BC. The only “foreign” find that was observed is a Spanish amphora, and it is said that it was sunk two centuries after the initial sea wreck. The site was used for photogrammetric survey and many amphorae were retrieved for analysis during 2015-2016. The figure below presents the location of Capo Sagro.

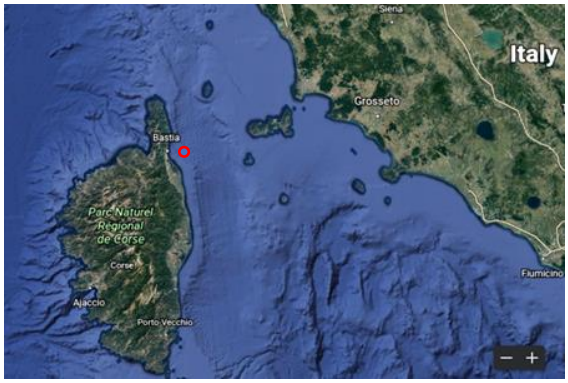


Figure 31. Approximate position of the Capo Sagro 3.

Implementation of the system:
As it was previously mentioned, ORB-SLAM 2 provides only three executables along with their setting files for three popular datasets, after it is built. Therefore, it was necessary to create another executable, enhanced by the map save function and modify the setting file, based on the calibration parameters as well as the AQUALOC’s environment. With respect to the context of images, i.e., several textureless areas, the images’ resolution and the capability of ORB detector/descriptor, it was found that a number of 2000 features was sufficient, in order for the system to do the initialization efficiently. At first, 500 salient points were selected to be extracted, but the system could not initialize. Then, a number of 1000 features was selected and the ORB-SLAM 2 ran. The initialization was performed after 68 seconds. Conversely, with 2000 features, the system achieved an initialization after only 10 seconds. Moreover, the first frames showed a small area of the amphorae, meaning that there was a high variability in images, in contrast with sandy regions where the pattern has low details. Also, the ROV performed both translation and rotation which is very important for the homography and fundamental matrix estimation, since frames with enough parallax should be used. Although, the algorithm is extremely fast at extracting features, it was not tested to run with more points as they could be incorrect. Besides, Mur-Artal et al. (2015) proposed the number of 1000 features for low resolutions (from 512×384 to 752×480) and 2000 features for higher resolutions. The tracking should rather be lost but precise than last longer with false results (false map points and frame detection). The camera’s calibration parameters were also added to the setting file, according to the calibration of Ferrera et al. (2018) as follows:

focal length → f_x=543.3327734182214, f_y=542.39877 pixels
principle point → c_x=489.02536042247897, c_y=305.38727 pixels
coefficients of radial and tangential distortion → k_1=-0.1255945656257394,
k_2= 0.053221287232781606, p_1=9.94070021080493e-05, p_2=9.550660927242349e-05, k_3=0
Frames per second (fps) = 20

It is noteworthy that ORB-SLAM 2 needs calibrated cameras but not necessarily undistorted ones. The use of undistorted images is time-consuming and hence, they cannot be used for real-time applications. Therefore, ORB-SLAM 2 takes into account the distortion parameters given in the configuration file, and “undistorts” only the keypoints instead of the whole image.

Before the implementation of the system on the dataset, a text file should be provided to process the images in the correct order according to their acquisition time. This file contains two elements, the timestamps and the names of the images. ORB-SLAM 2 is constructed in such

way that the input data should be named after their original timestamps, so that the former can make the association between them. For this reason, the images of AQUALOC were renamed according to their given timestamp and a text file was created with the above characteristics. Since the system was implemented in the same way as the TUM dataset, the timestamps given in nanoseconds, were converted into seconds and therefore, their precision in the main function of the source code was changed to 6 decimals (e.g., 1.54288371146347*10^18 ns → 1542883711.463470 sec). The algorithm's execution time was 18' 34''. Its mean tracking time was 65 ms and its median time was 70 ms. The tracking time includes the ORB extraction, the initial pose estimation and the tracking of the local map. As it was said before, tracking was working at a frame rate of 20 Hz, with the most demanding task, being the local map's tracking. The time can be reduced by decreasing the number of keyframes that are involved in the local map. Comparing the tracking time with the one of the TUM or EuRoC datasets, it could be deduced that the former was lower (40 ms). Tracking time depends on the number of extracted features, the image resolution, the relocalization procedure and the computer configuration. Although, in the AQUALOC dataset no relocalization was performed, the feature points and the resolution were higher. During the 3D map reconstruction, it was found that ORB-SLAM 2 could find a sufficient number of matches between processing frames (~ 500) despite the visual disturbances, the dynamic changes and textureless areas like the seabed. In the latter case, the system was able to find features because of small changes of intensity. However, due to these conditions many correspondences are spurious. The figures below represent the unique characteristics of underwater environment such as turbidity and backscattering which confine the typical process of SLAM.

Figure 32. Fishes in the vicinity of the camera

Figure 33. Backscattering on the amphorae.

Dynamic changes: In underwater environments, an important problem that occurs, is the marine wildlife, especially, in the presence of archaeological finds. Here, many fishes entered very fast in the vicinity of the camera and then went out immediately, increasing the noise and decreasing the number of salient points useful for the automatic extraction.

Backscattering: is an effect of white dots when using internal flash or strobes right next to the lens, which is getting worse when the water is filled with sand or plankton. Although, the system used for image acquisition has two strobes far away from the lens in both sides, this effect occurs several times, especially in the amphorae region. This may derive from the fact that the strobes were pointed towards them. Instead, they should be placed in such way that the object be illuminated with the edge of the "cone".

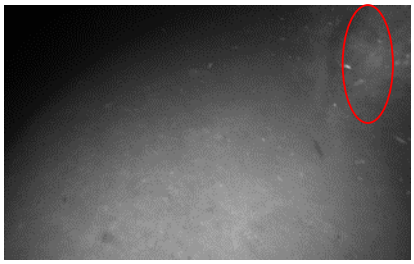


Figure 34. Sandy clouds.

Sandy clouds and turbulence: occurred at one time during exploration and were caused by the system's robotic arms (manipulators) that touched the seabed.

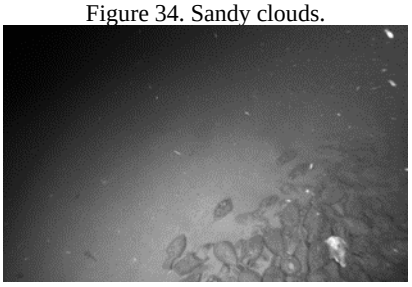


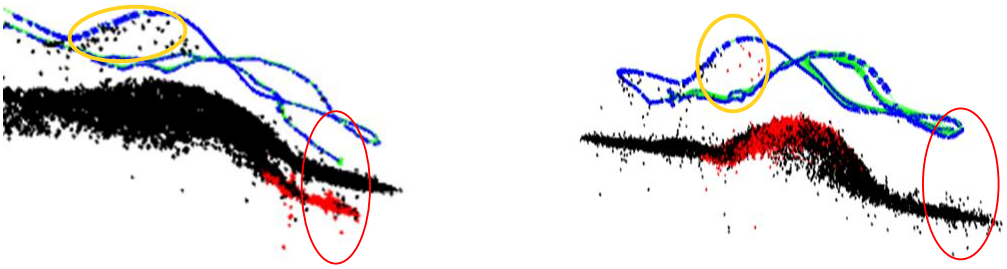
Figure 35. Turbidity.

Turbidity: occurs when water contains particles that cause cloudiness and muddiness. These particles are lighted more since light scatters. This can be caused by plankton and other microscopic organisms that got tangled with water.

During the navigation of the ROV, the camera performed several rotations and translations, and rarely approached and moved away from the object. However, the system, did not lose tracking (relocalization) in any of the above cases and the estimated trajectory closely followed the robot's motion, meaning that it can handle small rotations and is partially invariant to scale.

Figure 36. Archaeological site. Far exploration and in-close exploration.

It was also noticed that, while the algorithm was reconstructing the 3D map, there were duplicated map points resulting in an inconsistent map (producing two planes with different altitude). This is a result of the scale drift produced by monocular cameras, where depth measurements are not present. For this reason, ORB-SLAM 2 introduced a very important task, the global optimization which includes loop closures and pose-graph optimization, to handle the sequentially propagated errors and drifts, and to achieve global geometric consistencies of the global map along with camera trajectories. As soon as the system performed the global Bundle Adjustment to obtain an optimal solution and informed the user through the command line ("Map updated"), the duplicated map points were fused, and the accumulated error was corrected. Through full BA, the correction of keyframes was propagated to the non-updated keyframes that were inserted when the former was running, based on the spanning tree. The non-updated map points were also refined according to the correction of their reference keyframe. This led to a coherent result. From the figures below, one could also observe that the left image taken first, includes more wrong points (yellow circle), i.e., on top of the "hill" than the second one, that was taken later. This was a result of the map point culling task of ORB-SLAM 2, which is performed soon after the map point creation. Map points in order to be maintained, should satisfy two conditions: they have to be (a) trackable in more than 25% of frames in which is predicted to be and (b) observed from at least three keyframes (if more than one keyframe has passed from their creation). They can also be deleted after the keyframe culling and the local Bundle Adjustment which abandons spurious observations. In this way, the map is resilient to outliers.



(a) (b)

Figure 37. 3D map reconstruction and trajectory: (a) without loop closure. (b) with loop closure.

Based on the visual context of images, i.e., the examined underwater environment, it could be observed that the system generated several erroneous map points, some of them hovered over the archaeological site with amphorae. Wrong points could also be found around the hill, describing the amphorae, and the seabed. This arises from the visual degradation derived from the above underwater characteristics and the depth of the captured area, and the vivid marine wildlife, i.e., fauna and flora. During the exploration, many fishes entered in the camera's view while others laid down on the seabed and hence, the points that described them were useless. Furthermore, phytoplankton has been formed on the surface of the amphorae over the years and thus, they could not be reconstructed in their original form. Generally, the algorithm provided descent results of the 3D reconstruction and keyframe trajectory, considering the general geometry of the object and the ROV's positions, respectively. The system created 54741 map points.

Figure 38. Spurious 3D map points during reconstruction.

The generated keyframe trajectory consisted of 654 keyframes out of 11377 frames (5.75% of total frames). This was a result of the keyframe decision and culling, two very important steps incorporated in the tracking and local mapping threads of ORB-SLAM 2, respectively. During the first process, the system decides which frames will be considered as keyframes, based on some conditions, mentioned in the previous section. In short, a keyframe is inserted in the system only when its visual context changes. The second process detects unnecessary keyframes and removes them, in order to reduce the computational cost of Bundle Adjustment which is directly associated with them. In this way, a real-time and lifelong operation in the same environment can be achieved. During the implementation, it was noticed that the rate of the keyframe creation kept decreasing as they did not offer very new visual information and the biggest part of the scene was already reconstructed. In order to evaluate the performance of the system and the localization accuracy that this could obtain, the produced keyframe trajectory was compared with the groundtruth. These two trajectories were referred to a different coordinate system. The groundtruth was created using a conventional 3D reconstruction method, the SfM, trying to match every processing frame with all the others. This led to a fairly reliable trajectory, as many closed loops were present. According to Ferrera et al. (2018), the

groundtruth trajectory was scaled based on the pressure sensor measurements. It consisted of 569 frames (one frame for every 20 frames in the dataset) and the mean reprojection error of the 3D map points was 0.645 pixels. The keyframe trajectory produced by ORB-SLAM 2 was referred to an arbitrary coordinate system, where the scale was selected as the median depth of the initial map during the initialization procedure. Therefore, the trajectories should first be referred to the same coordinate system, to be comparable, i.e., a transformation matrix should be applied to the produced trajectory. This is a similarity transformation which has 7 DoFs (3 for translation, 3 for rotation and 1 for scale), since the two systems have different translation, rotation and scale. For this, the absolute orientation problem of Horn (1987) was leveraged. Since three points, known in both systems, provide 9 constraints, they are more than enough to estimate the 7 unknowns. The overall procedure can be summarized as follows:
- Given two sets of measurements in both systems, their centroids are calculated, using all the measurements so that, the coordinates of points are referred to them.
- For each pair of points in the two systems, the 9 possible products are calculated, e.g., if $r_l=x_l, y_l, z_l$ and $r_r=x_r, y_r, z_r$, their products are (x_l, x_r), (x_l, y_r), ..., (z_l, z_r).
- They are added up to obtain the sum products:

$$S_{xx} = \sum_{(n=1,n)} x_{l,i} * x_{r,i}$$
$$S_{xy} = \sum_{(n=1,n)} x_{l,i} * y_{r,i}$$
$$\vdots$$
$$S_{zz} = \sum_{(n=1,n)} z_{l,i} * z_{r,i}$$

- Afterwards, the 10 independent elements of N matrix are calculated, where the sum of diagonal elements equals to 0. The N matrix is:

$$N = \begin{bmatrix} (S_{xx}+S_{xy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{xy}-S_{yx} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}-S_{xz} & S_{yz}-S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix}$$

- They are used to estimate the eigenvalues of N which will later lead to the eigenvector calculation. The quaternion which represents the rotation is a vector with the same direction of this eigenvector.
- Next, the scale is computed as the ratio of the root-mean square deviations of the two sets of measurements from their corresponding centroids.
- Finally, the translation is the difference between the right centroid and the scaled and rotated left centroid.

In this way, neither approximation (i.e., a good initial guess) nor iterations are needed.
Horn's method relies on the translational part of the camera poses and not on the rotational. This derives from the fact that SLAM is a sequential algorithm and thus, rotational errors would appear as translational ones later (Salas et al., 2015).
For the evaluation of ORB-SLAM 2 in terms of both accuracy and robustness, two metrics were calculated, the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE). These two errors are strongly correlated (Sturm et al. 2012), so providing both will give a better understanding of the estimation quality. The ATE assume that the similarity transformation is first applied to the produced trajectory, so that both are referred to the same 3D coordinate system and with the same scale. The RPE does not need any alignment, only when the scale of the estimated trajectory is unknown, the former should be given. The first metric is well-suited

for measuring the performance of ORB-SLAM 2, whereas the second one can be used to measure the local and long-term consistency. The figure below illustrates the Absolute Trajectory Error and the Relative Pose Error.

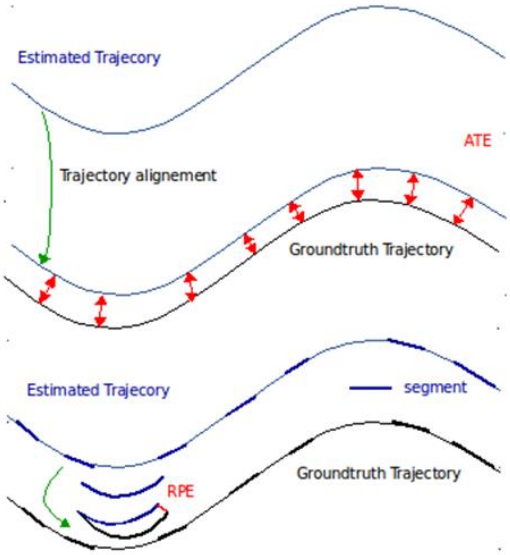


Figure 39. Illustration of ATE and RPE measurement.

Absolute Trajectory Error

The ATE measures the difference between associated points (poses) that belong to the estimated and the groundtruth trajectory. It provides a single number metric for both position and rotation estimation. It is also sensitive to the time the error occurs. For example, when an error happens at the beginning of the trajectory, it tends to give a larger ATE, than when it occurs at the end. This is illustrated in the figure below.

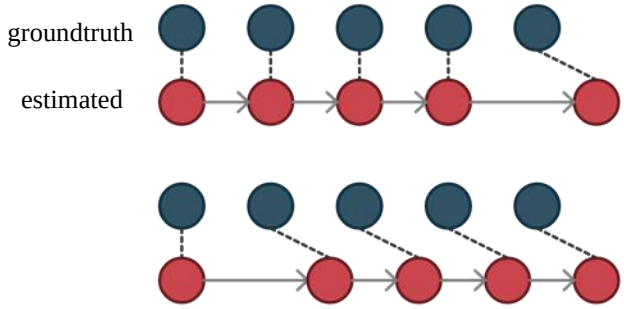


Figure 40. The error occurred in the pose estimation. (Top) at the end. (Bottom) at the beginning. The dark blue circles show the groundtruth positions while the red circles show the estimated trajectory. Dashed lines stand for the correspondence between the two

The association between them was achieved using their timestamps. The format of the generated trajectory text file was as follows: (timestamp) tx ty tz qx qy qz qw where tx, ty, tz are the 3D coordinates of the point in the world coordinate system (3D camera pose) and qx, qy, qz, qw, that give the orientation of the camera which is in form of a unit quaternion. The groundtruth trajectory file provided by Ferrera et al. (2018) includes the frame number, i.e., the number of the corresponding frame in the entire sequence. Therefore, to be compared with the keyframe trajectory produced by the ORB-SLAM 2, the frame number was renamed after its corresponding timestamp.

The script used, was provided by Mur-Artal R., the developer of the presented algorithm. The figure below depicts the two trajectories, choosing the generated to be the one most representative among the other executions.

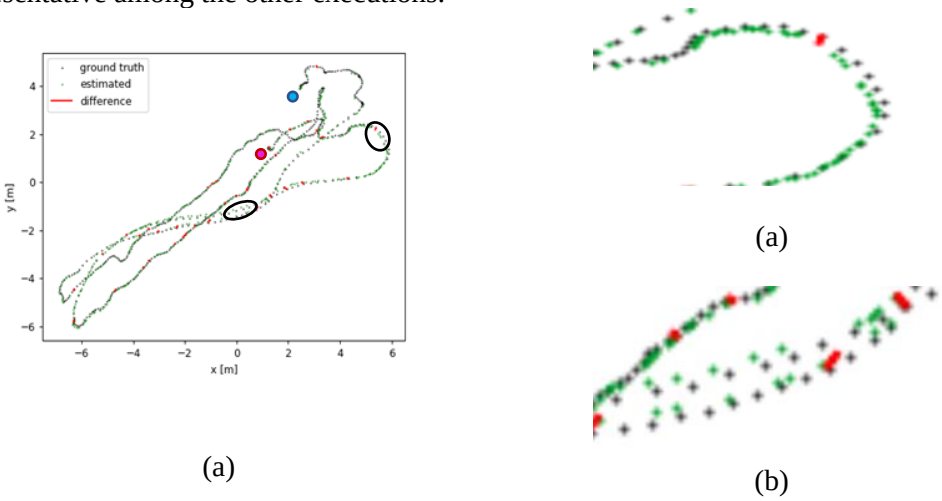


Figure 41. Plot of groundtruth (black) and estimated (green) trajectory with their differences (red).

From a graphical point of view and comparing the groundtruth with the produced keyframe trajectory, it could be deduced that the latter roughly followed the reference trajectory's geometry.

In many parts of the chart (previously presented figures), the drift was big especially in turns (b) and in regions where neither groundtruth nor estimated poses were very dense (sparse trajectory, (c)). Figure (a) represents the two trajectories. The reference trajectory is depicted as the dashed black line and the estimated as the dashed green line. The light blue dot illustrates the ROV's starting point and the magenta dot, the ending point. Figure (c) depicts two lines. The first line was formed as the ROV started its trajectory while the second one as it turned back (opposite directions). Although, ORB-SLAM 2 is able to detect and close loops (recognize revisited areas) and thus, correct the drift, the place recognizer cannot detect them from very different viewpoints, since many features are no longer visible. Therefore, it is not able to efficiently close the loop and correct the drift. In this case, the drift was also caused by the pure rotations that the robot previously performed.

Since ORB-SLAM 2 is randomized (non-deterministic), i.e., its initialization procedure is arbitrary in every implementation and hence, the keyframe poses are referred to a different coordinate system and scale, the AQUALOC sequence was run 9 times to find the median value of each metric, which represent accurately the behavior of the system. The estimated metrics of the absolute trajectory error of each execution and the associated pairs that were found between the two trajectories are presented below:

ATE	Execution	1 (33 pairs)	2 (32 pairs)	3 (27 pairs)	4 (35 pairs)	5 (34 pairs)	6 (43 pairs)	7 (24 pairs)	8 (43 pairs)	9 (29 pairs)	Median
RMSE (m)		0.075	0.100	0.089	0.351	0.090	0.209	0.289	0.394	0.252	0.209
Mean (m)		0.064	0.089	0.082	0.167	0.078	0.179	0.144	0.146	0.115	0.115
Median (m)		0.054	0.075	0.086	0.081	0.069	0.147	0.078	0.076	0.072	0.078
Standard Deviation (m)		0.039	0.048	0.035	0.309	0.044	0.107	0.25	0.366	0.224	0.107
Min (m)		0.023	0.033	0.029	0.021	0.024	0.049	0.032	0.024	0.006	0.029
Max (m)		0.153	0.238	0.149	1.408	0.195	0.427	1.308	2.50	1.328	0.238
Scale		1.529	1.516	1.517	1.513	1.50	1.426	1.487	1.50	1.51	

Table 2. Metrics of the ATE for each execution.

From the above metrics, it was noticed that the localization accuracy was slightly changed among the executions. This was caused by the arbitrary initialization and the associations (number and quality of corresponding images) that can be found between the estimated and the reference trajectory and thus, their comparisons. Since the initialization procedure was random, a different reference frame was selected at each time and hence, a different coordinate system was created. Therefore, from the table above, it can be seen that the scale was also different. In areas where the pattern has low details, the correct feature selection that would lead to a good matching and therefore, a reliable pose estimation, was more challenging. Based on the median value of the metrics, it was observable that the error was about 8 cm. This error stood for the drift producing over time during the SLAM process which was causing the estimated localization to slowly diverge from its true value. Monocular ORB-SLAM 2 uses Visual Odometry for pose estimation which is extracted from the difference of the current frame to the previous one. This leads to an incremental change of the camera pose and thus, to accumulative errors. Although ORB-SLAM 2 has the ability to detect and close loops as well as perform global optimizations to minimize the reprojection error caused by the accumulated drift (translation, rotation and scale), this can still occur especially when the camera performs pure rotations and large loops from very different viewpoint are detected. ORB features provide good invariance to illumination and viewpoint changes. However, ORB-SLAM 2 uses only local appearance-based features that are not so robust to large viewpoint changes since the same surfaces are no longer visible.

Relative Pose Error

The RPE computes the relative motion between pairs of timestamps between the reference and the estimated keyframe trajectory. This metric is independent of the reference frame but if the map scale is unknown, a scale alignment should be first applied. Here, in order to measure the RPE, the information of scale was added from the result of ATE. The RPE is less sensitive to the time the estimation error occurs, due to sampling the trajectory over smaller segments (Zhang et al. 2018). In the case of evaluating the consistency of Visual Odometry, a fixed window size can be used, where each pose of the estimated trajectory is associated with a later pose according to this window size and unit, e.g., seconds. Selecting pairs that are spaced by a certain distance, e.g., number of seconds along the trajectory, gives different meanings. In particular, pairs that are spatially close (close in time) indicate the local consistency, while pairs with longer distance show the long-term accuracy. Moreover, one could average over all possible time intervals to compute the RMSE, but this would lead to a computational complexity that would quadratic to the trajectory length. Therefore, several window sizes were selected to compare their translational and rotational errors, both RMSE and median, since the latter is less sensitive to outliers (Sturm et al. 2012) as the squaring process “gives” higher weight to very large errors. With the increase of seconds elapsed, the matching pairs detected were decreased. The most representative trajectory of the previous executions was chosen to be measured.

RPE / Time intervals	10 s	20 s	50 s	90 s	150 s	200 s	250 s
Translational RMSE (m)	0.073	0.085	0.395	0.309	0.283	0.397	0.459
Translational median (m)	0.028	0.041	0.041	0.034	0.034	0.034	0.033
Rotational RMSE (deg)	1.42	1.15	1.98	1.79	1.96	3.16	3.55
Rotational median (deg)	0.78	0.62	0.60	0.66	0.58	0.58	0.55

Table 3. Metrics of the RPE.

From the table presented above, it could be noticed that, as the time between the selected keyframe pairs, i.e., the length of segments, increases, so does the RMSE. This reflects the drift which is accumulated over time. One could also see that the median error, which is more robust to outliers than the RMSE, had been maintained almost constant regardless of the time intervals. This indicated not only the local consistency that the algorithm could achieve, but also the global coherence.
The RPE was slightly greater than ATE as it assessed both translational and rotational errors.
From the above metrics, it concluded that the localization accuracy of the system was medium to high with respect to the environment and the available data. The overall accuracy and robustness depend on many factors that can be divided into two main categories: the environment and the ORB-SLAM's 2 performance.

Underwater environment:

The underwater environment has many limitations that hamper the typical process of ORB-SLAM 2. Dynamic changes constitute one of the most important difficulties when dealing with underwater scenarios. The marine flora as well as the fauna present in the examined dataset lead to image visual degradation. This degradation resulted in a rough feature extraction and therefore matching process, which are the primary steps in the SLAM procedure. Hence, it caused wrong correspondences which passed through all the ORB-SLAM's 2 threads and provoked an incorrect pose estimation even if many of them got discarded from the system, e.g., during Bundle Adjustment. Hard underwater conditions like blur, scattering, turbidity etc. which produce low contrast images, also contributed to error existence, since the algorithm either faced difficulty in extracting feature points or could extract feature points but not associate them correctly, because they changed position very fast. The wrong data association influenced the accuracy of pose estimation and the reconstructed map. Although textureless areas like the seabed had spots of different colors that can aid the process, there were many regions with extremely low variability making the feature extraction and matching challenging.

ORB-SLAM 2:

ORB-SLAM 2 is based on Visual Odometry which estimates the relative camera poses instead of the absolute ones. Although, VO performs local optimization to refine the frame positions, an incremental error in the measurements is appeared. The trajectory drifted to translation, rotation and scale, since neither depth nor other external measurements could be taken into account. If the system considered only the approach of VO, the error would be extremely large and thus, prohibitive. However, due to its capacity to detect and effectively close loops as well as perform global optimization and back-end Bundle Adjustment, the accumulated error was maintained low and global consistency was achieved, regarding the characteristics of the environment. One more possible cause is that the system reduces the global optimization to a pose-graph optimization which discards the sensor measurements and distributes the loop closing error along the essential graph.

Other parameters:

Furthermore, it should be pointed out that the reference keyframe trajectory was generated from the state-of-the art SfM Colmap library (Schonberger et al. 2016). This approach included a post-processing 3D reconstruction to compute the camera poses by matching all the acquired images, composing a sequence and leveraging the loop-closure information. Although, the given 3D reprojection errors and the statistics of the 3D reconstruction were decent, the produced keyframe trajectory could not be considered as a perfect groundtruth.

Capo Sagro 3	
Number of used images	569
Number of 3D points	251620
Mean tracking length (m)	7.4
Mean reprojection error (px)	0.645

Table 4. Statistics of the 3D reconstruction.

In land, the acquisition of groundtruth is easily derived from the GPS. This is unfeasible for underwater scenarios where the signal immediately drops-off. Solution to this, is the integration of acoustic sources which are rather very expensive and therefore, less attractive.

Additionally, it was deduced that ORB-SLAM 2 is not capable to deal with fisheye lenses. It is designed to process only pinhole or near pinhole cameras. During initialization, the initial map is created computing two models, a homography for planar or near planar scenes and a fundamental matrix for non-planar scenes; and the relative camera motion is recovered. This is suitable only for conventional cameras and thus, ORB-SLAM 2 cannot exploit all the information from a fisheye image. This is caused by the fact that the pinhole camera model projects the 3D points into a 2D plane. Consequently, a new camera model with further adjustments should be applied, because even if the projection model would be provided, the existed pinhole model would present a suboptimal performance for a FoV greater than 120 deg (Liu et al. 2019).

B. Zürich Urban MAV Dataset

Overview of the dataset:

The other dataset that was used to evaluate the ORB-SLAM's 2 performance was a subset of the Zürich Urban MAV Dataset developed by Majdik et al. (2016). The calibrated images (number of 81169) shown the facades of the buildings in the streets of Zürich were recorded by a rolling shutter GoPro Hero 4 camera embedded on a Fotokite quadrotor (MAV). The drone was flying in a downtown area of Zürich at low altitudes (5-15 meters from the ground) in the center of the streets. The images were time-synchronized in high-resolution (1920×1080×24 bits) and were obtained at a frame rate of 30 Hz. The dataset also provides GPS data, IMU measurements, ground-level Google-Street-View-images, timestamps expressed in microseconds and groundtruth data in the WGS 84 coordinate system. The groundtruth trajectory derived from an accurate photogrammetric 3D reconstruction by sub-sampling the data at 1 fps and recording them in such way to include loop-closures. Also, additional 2D control points were manually marked, to achieve a coherent 3D reconstruction. The GPS position was used as initial, to perform Bundle Adjustment iteratively (Mean reprojection error → 0.216531 px). The format of the groundtruth was the following: frame ID, X, Y, Z, omega, phi, kappa, Xgps, Ygps, Zgps, where the frame ID declared the number of frame in the dataset, X, Y, and Z are the groundtruth camera positions in WGS84/UTM zone 32N system, omega, phi and kappa angles defined the orientation of the camera, and Xgps, Ygps, Zgps were the coordinates measured by the GPS in International WGS84 GPS system. The omega, phi and kappa angles can represent the yaw, pitch and roll angles which define the rotation of the MAV body. Omega angle defines the rotation around X axis, phi angle defines the rotation around Y axis and kappa explains the rotation around Z axis.

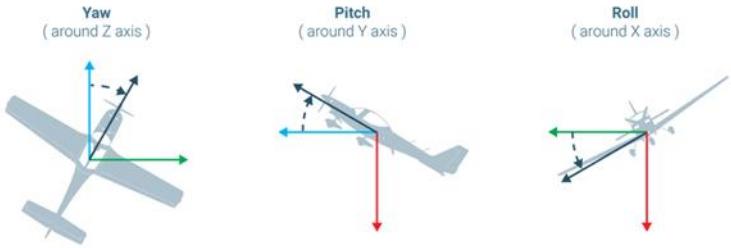


Figure 42. Yaw, Pitch and Roll angles of a drone (Pix4D Documentation, Yaw, Pitch and Roll angles).

Although, GPS receivers provide accurate localization, their accuracy and reliability totally depend on the number of visible satellites and thus, their result cannot be considered as groundtruth. In urban canyons, i.e., big cities like Zürich, the GPS signal is either shadowed by the surrounding buildings (violating the direct signal's travel from the satellite to the receiver, i.e., multipath) or not available at all. The original length of the dataset is 2 kilometers.

Figure 43. The MAV Fotokite.

RMSE geo-location error (m)	
RMSEx	2.22
RMSEy	3.76
RMSEz	5.46

Table 5. RMSE error (Majdik et al. 2016).

Description of area of interest:

The sequence selected from the entire dataset represents two streets in the center of Zürich in Switzerland that were not very crowded, i.e., there were not many passengers or cars passing by. The streets were characterized by tall building and few trees.

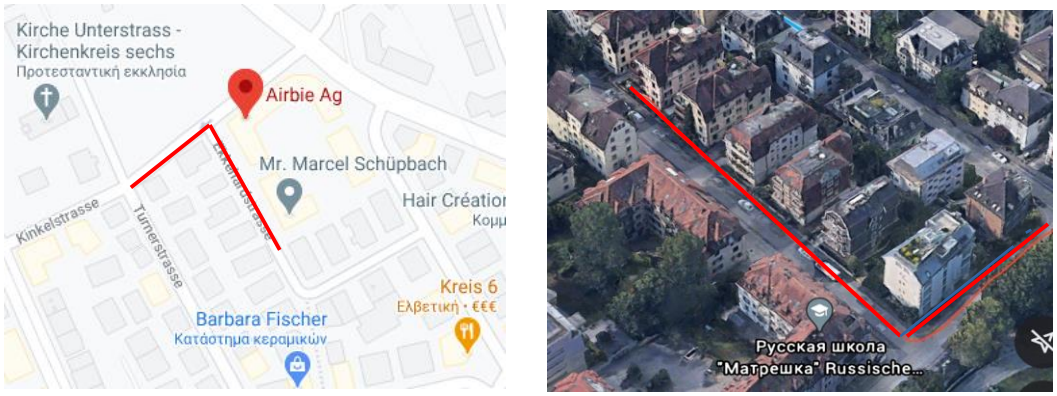
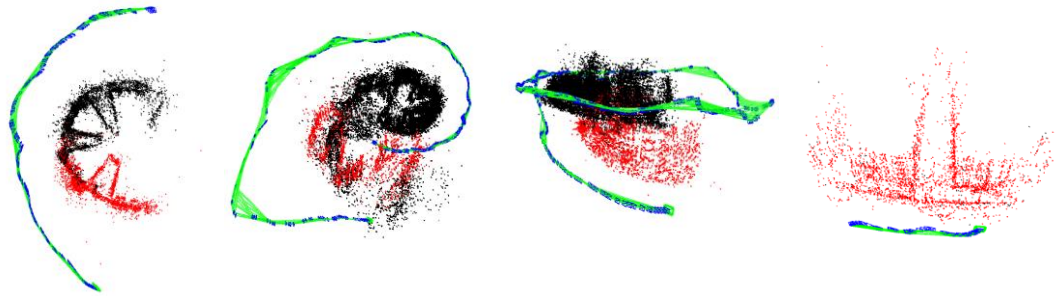


Figure 44. The bird-eye view of the urban test area. The red lines mark the selected sequence.

Implementation of the system:

Due to the original length of the dataset, i.e., the huge number of images and therefore, the long duration of the implementation, a subset of those images was selected. The selection was made in such way that the dataset contains rotations, repeated patterns on the building facades and dynamic objects, i.e., people walking down the streets. Specifically, 7714 images were

extracted from the dataset, covering an area of 190-meter long. At first, the configuration file that had to be provided, was modified according to some parameters, such as the very high image resolution (1920×1080×24 bits), the calibration parameters, the fish-eye lens and the nature of the examined environment. The first execution was performed, resizing the images so that the new images' shape was the 30% of the originals (both width and height). The fps was set to 30 according to the acquisition time of images and the number of extracted features to 2000. Although the initialization was achieved in 2 seconds, the keyframe trajectory as well as the 3D reconstructed map completely failed. Also, the execution time was excessively big. The figures below represent the result of ORB-SLAM 2.



(a) (b) (c) (d)

Figure 45. Keyframe Trajectory and 3D map construction with the decrease of original image resolution.

The result was far from optimal. Although, at the beginning it worked well with respect to the environment, the error was accumulating over time. During exploration, it was noticed that the reconstructed map was locally consistent, and the temporal result resembled the real scene (Figure 44d). However, as the distance grew the drift became bigger, leading to global inconsistency (Figure 44a, 44b, 44c). The estimated trajectory and the map seemed to follow a circular motion while they should represent a straight line. This resulted in one part of the scene being inserted into another (Figure 44b). The Figure 44c was captured at the end of the process and one could detect the discontinuity between the two parts of the scene (the two streets). Although the two roads were not at the same altitude and the drone slightly changed its flight altitude, the system could not accurately reflect it, in both the map and estimated trajectory, instead it presented a bigger difference. This disastrous result derived from a variety of parameters such as the decrease of image resolution as the image quality was shrunk and therefore, the feature extraction and matching were degraded. The other factors that influenced the accuracy and the robustness of the performance will be analyzed below. Since the solution was far from reality, no further computations and evaluations were made.
Although, the algorithm was tested for 20 fps and 3000 extracted features where the execution time was decreased, the result was still confused.
After some tests, the input images were resized in such way that their new shape was the 90% of the original. The frames per second were set to 30 and the number of feature points, to 2000 as Mur-Artal et al. (2015) proposed. The other parameters were set by default. The calibration parameters were given by the creators of the examined dataset and are presented below. The calibration was performed with OpenCV library, using a checkerboard of nine squares wide and seven high, with a square size of 2.45 cm. The parameters were estimated based on the pinhole camera model. The images did not get undistorted as they would add more computational cost to the process.

Calibration parameters

focal length → f_x=893.39010814, f_y=898.3264 pixels
principle point → c_x=951.1310043, c_y=555.1335007 pixels
coefficients of radial and tangential distortion → k_1=- 0.2805251300000,

k_2= 0.1158064130000, p_1=-0.0009843367850, p_2=0.0001584792480, k_3=-0.0270215034000
Frames per second (fps) = 30

The timestamp of each image was converted from microseconds to seconds (1 sec → 10^-3 microseconds) so that can be run in the same way as the TUM dataset does. Also, the images were renamed after their timestamp and a text file including both names and timestamps was generated. Therefore, their precision in the main function of the source code was changed to 7 decimals (e.g., 1519783350 microseconds → 151.9783350 sec). The algorithm execution time was very long, about 38' 58''. The mean and median tracking time was 170 ms. Tracking includes the feature extraction and matching, the camera pose estimation and its refinement, the relocalization in the case of failure and the keyframe selection. Therefore, its time depends on the image resolution (high image resolution means high quality), the necessity of a relocalization, the number of extracted features and the computational power. In the current implementation the input images were not greatly reduced from their original resolution so that no useful information got lost. This increased the processing time. Although the drone performed a rotation, due to its low speed and soft movements, the system could process all the frames and no tracking failures occurred. The figures below depict the examined environment. It is observed that the images captured by a fisheye lens are fully distorted. The Figure 45d shows the drone's propeller getting in the field of view of the camera. However, this did not affect the feature extraction and matching process.

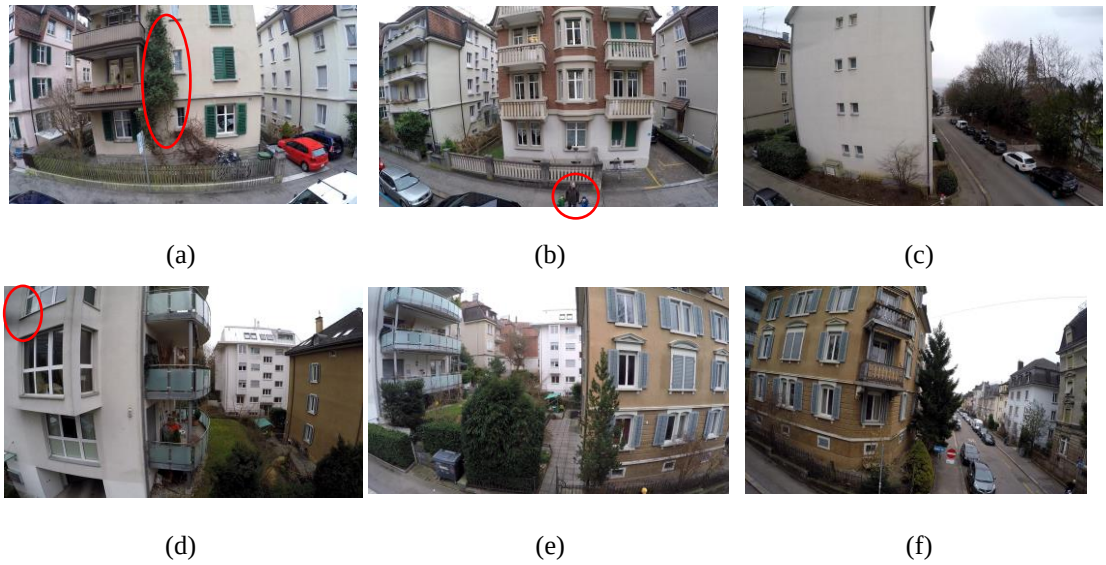


Figure 46. The test environment with its characteristics. (Top) the first street. (Bottom) the second street with lower altitude.

During the 3D reconstruction it was noticed that the system could find a lot of matches between two consecutive frames, most of them being correct. In the case of repeated patterns on the building facades, the algorithm did not find feature points, because the matching procedure could not be crowned with success, since there were many possible descriptors for one single point (the area around the point was very similar). Additionally, in the case of people walking down the streets, the ORB-SLAM 2 did find salient points and triangulate them, but soon after their creation, some of the latter were deleted by the "map point culling" procedure. Conversely, in the case of people who stayed still, the system created map points that are incorrect. Also, it generated points that described trees, cars, and other urban equipment such as garbage cans and signpost. As long as map points described stable objects, they could be considered correct and used for camera pose estimation as constraints. Points that described movable objects such as leaves and birds that entered in the vicinity of the camera or objects that were reflected by windows (e.g., furniture), were considered as noise and their maintenance interrupted the final result. In general, the triangulated map points described well the real scene while the surfaces

and the planes of the buildings were properly defined. The relative depth of the mapped areas seemed respectable according to the real world. From the reconstructed map, one could observe that there were many duplicated map points that were never fused, with respect to the environment. This led to a very confusing result disrupting the coherence of the map. As it was mentioned in the previous chapter, the mapping thread process new keyframes and performs local Bundle Adjustment. After, the loop closure procedure takes place and searches for loop candidates for the current keyframe in the database. This is achieved by the estimation of a similarity transformation, which is the only way to be aware of the drift accumulated during exploration and correct it. Here the drift occurred in 7 Degrees of Freedom since, one single camera was used for the recording. The dataset did not include any loop since the MAV has never returned to a previously revisited area. Therefore, there was not a way to correct the cumulative error and fuse duplicated points by closing loops during exploration. Moreover, the global BA was never called. In the loop closing class the conditions were never met to detect a loop and launch BA and hence, neither keyframes nor map points were optimized. Similar to the first execution, the local map was consistent while the system could not maintain the overall consistency of the full map, i.e., the road was presented curved instead of linear as it should be. However, it showed better results than the previous attempt. The figures below present the reconstructed 3D map as well as the keyframe trajectory during exploration and at the end of the process. Red map points contributed to the current pose estimation while the others were colored black. The camera poses that colored blue stood for the previous camera positions and green, for the current one. Keyframes that shared map points were connected with green edges (graph).

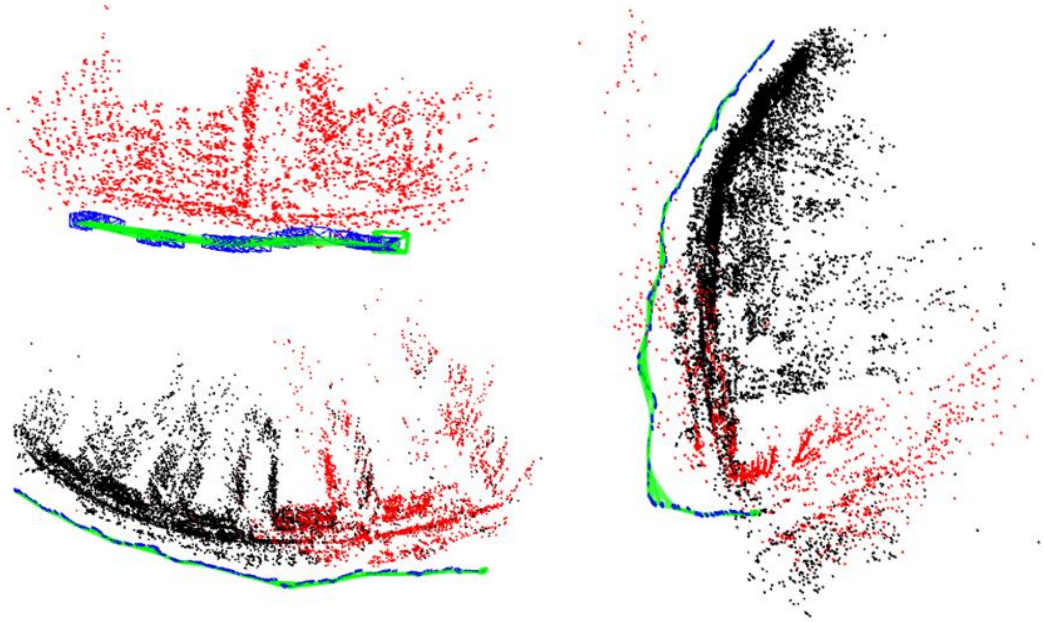


Figure 47. The generated 3D map and keyframe trajectory.

The generated keyframe trajectory consisted of 372 keyframes out of 7714 frames (4.82%). As mentioned above, the system based on the keyframe selection and culling procedure, significantly reduced the number of processed keyframes. This was also noticed during reconstruction where the algorithm created keyframes and afterwards, many of them got deleted. In a nutshell, ORB-SLAM 2 inserts keyframes as soon as possible with respect to some conditions and at the end, keeps only those which offer new visual information (discards the unnecessary ones). This capability enhances its accuracy, as the mapping is not tied to framerate and the computational cost of the optimization is decreased. Besides, the examined dataset was recorded at 30 fps and the execution was performed using the same fps, which means that every 30 frames the visual context was slightly changed. Therefore, ORB-SLAM 2 kept approximately one keyframe per 30 frames. A problem that was observed when the produced

keyframe trajectory was plotted, was that it was different from the one which displayed during the process and had different scale comparing to the 3D map. Therefore, the localization accuracy could not be evaluated using the groundtruth. Besides, since no loop closures carried out, the error was getting very big as the traveled distance increased.

4.2.3 Implementation of the Visual Map

As mentioned before, during the 3d reconstruction, i.e. the creation of the sparse point cloud, one cannot extract an adequate information about the reconstructed environment due to the sparsity and the large mapped area. The images that are displayed in a separated window, go by in a rapid manner, one after the other, making difficult the memorization to the user. The long-term aim of this diploma thesis is the integration of the visual map inside the ORB-SLAM 2 so that the proper is created along with the 3D map and keyframe trajectory. However, at first some efforts were made, separately from the execution of the algorithm.

The created keyframes covered the entire region of the archaeological site, since the redundant keyframes were discarded based on some conditions mentioned above. Therefore, they were sufficient to illustrate the visual map completely.

As soon as the ORB-SLAM 2 was executed for the AQUALOC dataset, three files were generated. The first one contained the 3D coordinates of the map points, the second one contained the keyframe trajectory, i.e., the translation and orientation of the camera, and the third file was the "homography.txt", which included the keyframes produced by the ORB-SLAM 2 and the correspondences between two consecutive keyframes. At first, the proposed approach was tested for 4 keyframes in order to check how it works. Two tests were performed, the first set of keyframes depicted an area with high variability (i.e. amphorae) with the camera performing only translation, while the second one showed a region with few amphorae presented in the borders with the camera performing both translation and rotation. In both cases, disturbances (fishes in the vicinity of the camera) were present. The two final images are presented below.

Figure 48. Final image "mosaic" of a textured area with translation.

It was observed that the image alignment was successful. Taking a look at the borders of the images one can notice that the amphorae fit well from one frame to another. However, the seams in the overlapping area of the images are very present during the alignment. This is caused by the exposure differences. During acquisition, the images often differ in color and brightness due to environmental changes and camera exposure time. Here, the unique conditions of the environment, the fishes and the turbidity which is also presented in the above keyframes hamper

the acquisition of images with the same characteristics. The process time was 37 seconds (second approach). The corresponding points were plotted on two of the keyframes to check their reliability.

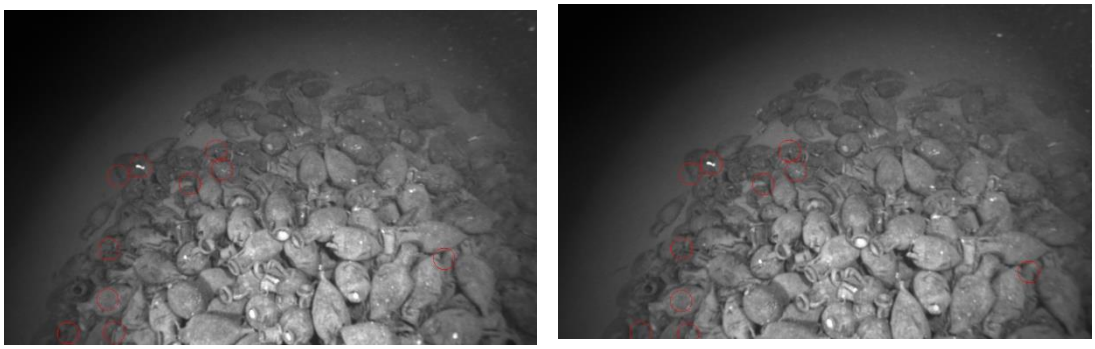


Figure 49. Corresponding feature points of the ORB-SLAM 2 between two sequential keyframes.

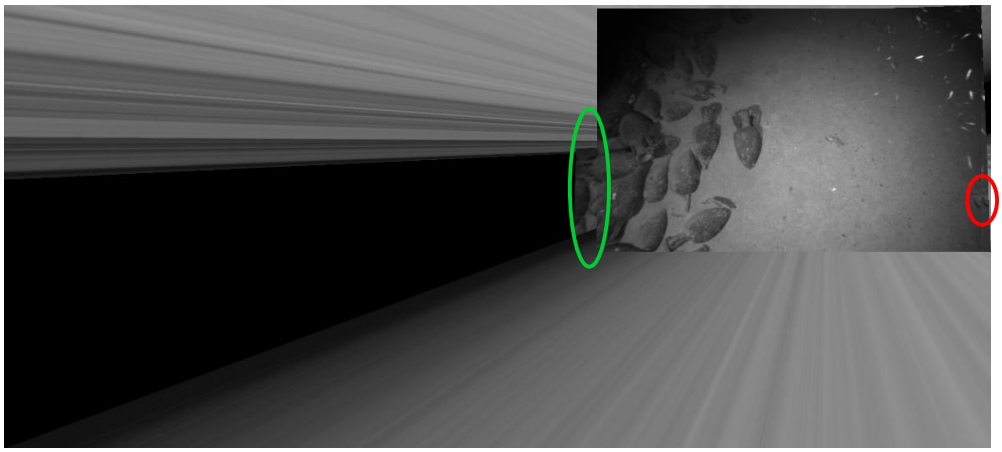


Figure 50. Final image “mosaic” with pure camera rotation and translation.

Based on the second experiment, it was noticed that the approach could not process those frames correctly. The first two images were well aligned (green circle). However, they were not properly aligned with the third one (red circle). Although, most of the outliers were abandoned and the homography was optimized, still the presence of noise could lead to this misalignment (recall). In order to investigate the reason behind this and check if the correspondences from the ORB-SLAM 2 as well as the ORB operator were correct, they were plotted on the keyframe pairs. The RMSE and the recall metrics were also computed and gave a value of 3 pixels and 0.77, respectively.

The last two images were never aligned, and the result of the output image was the above. This was caused by the angle that was formed between the image plane and the reference image plane due to the pure camera rotation. The closer the angle is to 90 degrees, the farther away the projected points will be (infinity). This is illustrated in the figure below.

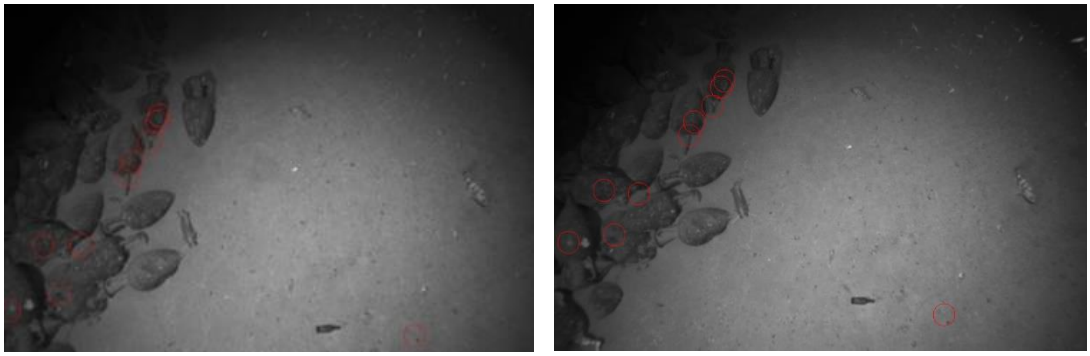


Figure 51. Corresponding feature points of the ORB-SLAM 2 between two sequential keyframes.

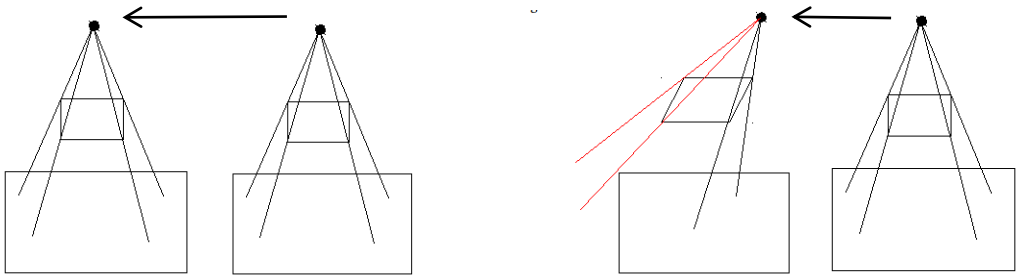


Figure 52. Projective geometry between two planes.

Since the approach relies mainly on image registration, visual results are not sufficient to appreciate the work. Therefore, some objective metrics were used to evaluate the accuracy of the homography estimation.

1) RMSE

Having the coordinates of a pair of matched points (x_i, y_i) in the first image and (u_i, v_i) in the second one as well as the homography matrix relating the two images, the transformed coordinates of the second image in the first one are computed as:

$$(U_i, V_i, 1) = H\ (u_i, v_i, 1)$$

The distance (D) between a point in the first image and its corresponding point after transforming in the second image is:

If N is the number of total correspondences, the RMSE value is:

$$RMSE = \sqrt{\frac{1}{N} * (\sum[(U_i - x_i)^2 - (V_i - y_i)^2])}$$

Considering the matches produced by the ORB-SLAM 2, the RMSE in both cases and between all the keyframe pairs was very low, around 2-3 pixels.

2) Recall

This evaluation criterion is based on the number of inliers and outliers obtained from the image pair and is given by the following equation:

recall = Number of correct matches/total number of correspondences

The recall value for these keyframe pairs was around 0.76. Although it is smaller than 1 (i.e., the ideal value), it is close to it.

Another test using 23 keyframes was conducted in about 1.12 minutes using the faster approach. The image composition is presented below.

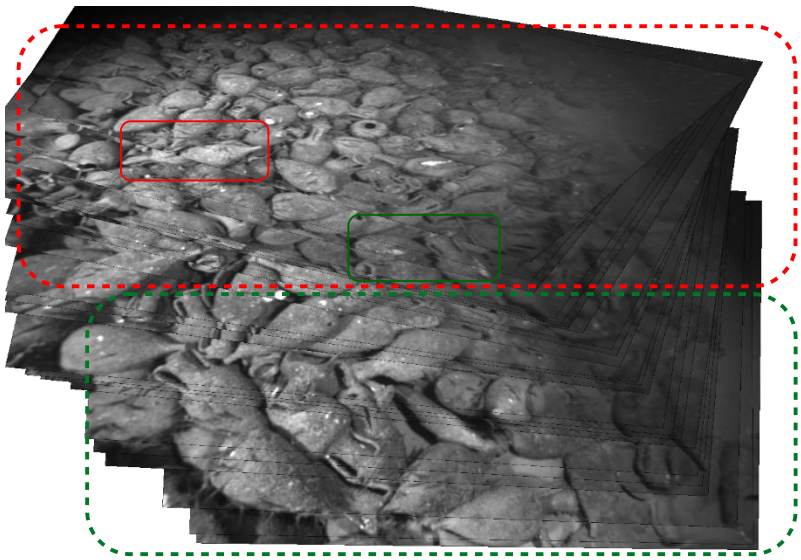


Figure 53. Final image "mosaic" consisting of 23 keyframes.

From the above figure, it was deduced that the final "mosaic" was not totally well-aligned, but only for the first 16 images (green). Although after those images the "mosaic" showed several misalignments (red), there were parts, where the alignment was successful. This was a result of different factors, such as the accumulated error in the homography estimation, the non-planarity of the reconstructed scene, the camera rotation and the distribution of feature points. More specifically, the homographies with respect to the reference image are estimated through a consecutive multiplication, as mentioned in the Chapter 4.1. Therefore, the errors existing in the homographies are transferred from one homography to the next one. Furthermore, the assumption for planarity was made without implying that the object is exactly planar, instead it has extreme relief. However, one can realize whether the area is covered or not.

In the last experiment, the approach was tested using 70 keyframes from the "finaltracks.txt" file and the transparent result was the following.

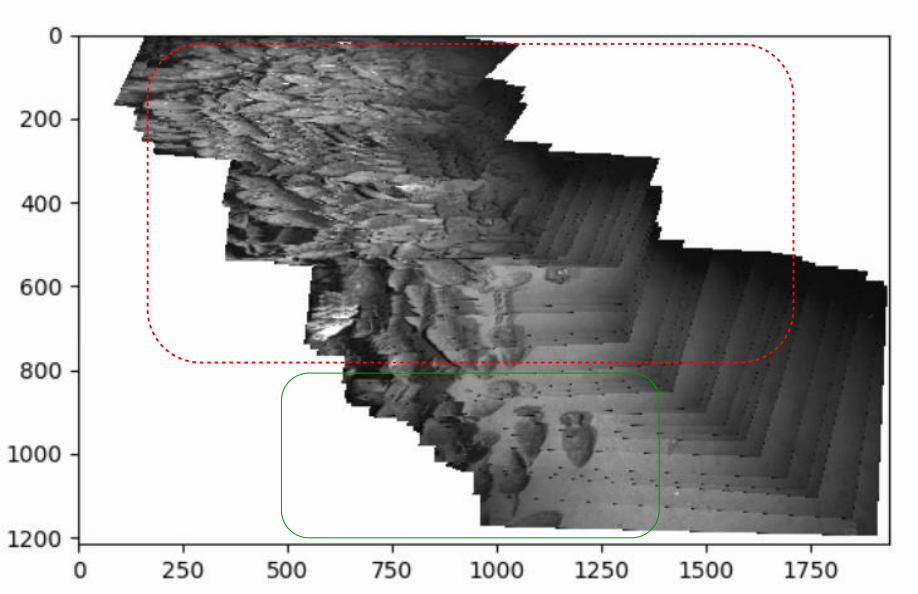


Figure 54. Final image “mosaic” consisting of 70 keyframes.

From the figure above, one could notice that as the images increased, the alignment error was accumulating. This resulted in inconsistencies between images that were more present after the first 10 keyframes. To ensure that the correspondences in the misaligned images were correct, the program ran using only those keyframes. From the figure below, one could see that the alignment was correct.

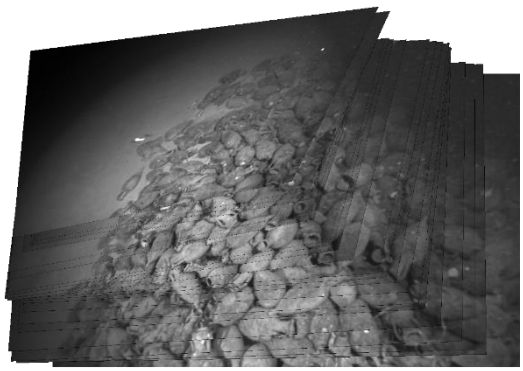


Figure 55. Final image “mosaic”.

Also, it was observed that the images could not fit inside the final image, meaning that the selected dimensions should be modified. Initially, the dimensions of the final “mosaic” were chosen to be the double of the images’ size.

The RMSE calculation as well as the plotting of the correspondences were performed for all the experiments. The matches were correct with a mean RMSE value of 3 pixels and the recall was around 0.75.

All the above experiments were also conducted using both approaches. Based on the execution time, it was deduced that the second approach was much faster that the first one especially, when the images used are increased.

	Keyframes	Approach	1st	2nd
Time (s)	4		110	37
	10		1345	78

Table 6. Execution time of two methods.

Furthermore, the number of keyframes used for the composition affect the execution time differently depending on the approach. Precisely, the growth rate of the execution time was lower in the second approach than the first one.

The proposed approach was tested with all the keyframes produced by the ORB-SLAM 2, but as soon as an error occurred (the points were projected too far away from the reference plane), it was transferred to all images and the result can never recover.

5. Conclusions and Future Work

This diploma thesis focused on the development and implementation of a Monocular Visual SLAM system which can achieve real-time performance. An extensive literature review was performed to find and evaluate a suitable SLAM algorithm for the implementation part. The ORB-SLAM 2 developed by Mur-Artal et al. (2017) was found to be the best SLAM system currently available. It is a feature-based algorithm which performs map reconstruction over selected keyframes with estimation of camera positions after every recall in dataset, creating a 3D keyframe trajectory as well. It consists of three main steps: tracking, local mapping and loop closing that run in parallel and use the same feature points, resulting in a significantly decrease of computational cost and an efficient and reliable process. It makes use of ORB detector/descriptor which is an order of magnitude faster than other popular algorithms such as SURF and over two orders faster than SIFT. This thesis provides an in-depth explanation of ORB-SLAM's 2 steps, describing the used techniques and the overall structure of the system. This algorithm generates a sparse and colorless point cloud which is deleted soon after its completion and a keyframe trajectory which includes the 3D position and orientation of keyframes, a sample of the input frames.
The algorithm stores all the final map points in a variable called GetAllMapPoints. Here, a function that saves the 3D map reconstruction into a PCD file was developed, in order to visualize and evaluate the result. This, in combination with the extrinsic parameters (trajectory) that are estimated, can also be used for building a 3D dense point cloud. From the sparse point cloud, one cannot embed critical information of the environment. The map is created based on a subset of frames, called keyframes. The keyframes encompass an intuitive summary of the environment with a high pose accuracy and rich information of the covisibility. Therefore, the sparse point cloud can be used as an excellent initial guess on which a denser map can be built. A denser map can be applied to high-level applications, such as obstacle recognition, navigation in a pre-built map, building 3D reconstruction etc.
Furthermore, due to this sparse result, a 2D incremental visual map was developed, with the aim of offering a more informative view of the examined environment to the user and therefore, its further analysis. It consists of keyframes produced by the ORB-SLAM 2 system and exploits two very important tasks of the algorithm, the feature extraction and matching. The proposed approach also gives the option to save the generated map as transparent.

5.1. ORB-SLAM 2 system

To test the performance of the system and evaluate it in terms of localization accuracy, two datasets in two different environments were used. The first one captures an underwater archaeological site of a shop wreck with a load of amphorae located at a depth of 380 m, while the second one shows two urban street scenes.
From the first experiment it was found that a number of 2000 feature points was appropriate in order to achieve a fast and good initialization (10 seconds). It was also tested for 500 and 1000 points but neither of them confirmed the above two conditions. For 500 features, the system could not initialize while with 1000 features, it did make an initialization but only after 68 seconds. Consequently, the initialization procedure depends on the number of extracted points and hence, the image resolution as well as the complexity of the scene, e.g., textureless sand areas. One should change this parameter regarding the examined environment. The algorithm which was executed based on the TUM dataset's source code, could not process the frames with timestamps given in nanoseconds, and the latter had to be converted in the same format as the TUM's timestamps, i.e., seconds. ORB-SLAM 2 needs small "gap" between the timestamps, to efficiently start the process. For example, if the following timestamps 1542883711463470000 and 1542883711513470000 in nanoseconds are not converted, the system considers them as seconds and perceives a large difference, meaning that it will process a frame at every 50000000 seconds which is unfeasible. Otherwise, the difference is barely noticeable, and the system would process at every 0.05 seconds. The mean tracking time was

65 ms and the overall execution time, 18’ 34’’, i.e., is twice the original duration of the video. Although the time could reflect a real-time performance, it can still be decreased by reducing the number of keyframes which are included in the local map, since tracking the local map is the hardest procedure in terms of computational cost. The tracking time depends on the number of extracted features, the image resolution and the relocalization procedure. During exploration, the SLAM system did not lose tracking despite the visual degradation and the special characteristics of the underwater environment. This is caused by the big number of processing frames per second that were used. This results in less movement per frame and thus, allowing a more robust tracking. It was also noticed that ORB-SLAM 2 was able to extract and match a sufficient number of feature points between frames but many of them were wrong, based on the final 3D map. This is mainly caused by the dynamic objects, i.e., fishes getting in the field of view of the camera, that were present in the dataset.

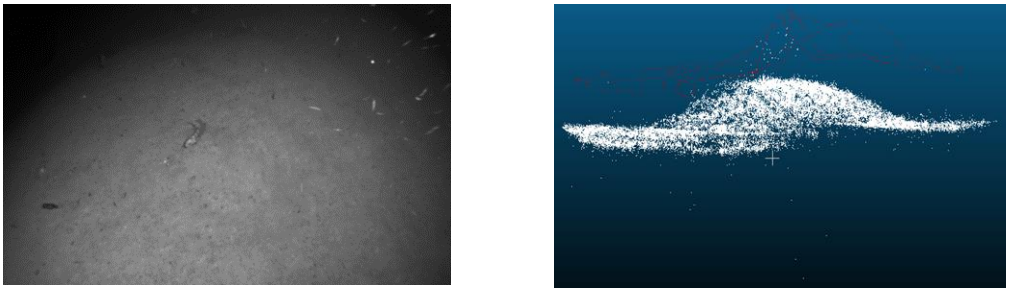


Figure 56. Erroneous map points in the final 3D reconstructed map due to moving objects.

More precisely, many points describing these dynamic changes, change their positions due to the object movement and hence, their maintenance for matching leads to wrong data correspondence. The incorrect data correspondence degrades the localization accuracy as in this case. Therefore, the algorithm can be combined with Deep Learning techniques, especially real-time semantic segmentation networks in a different parallel thread which merges the segmentation with a moving consistency check method, to filter out the dynamic objects of the scene. In this way, the influence of moving objects in pose estimation can be reduced. Such an approach has been investigated and developed by many researchers, such as the DS-SLAM introduced by Yu et al. (2018). Nevertheless, the final map was very close to the real scene considering the optical result. Further quantitative evaluation can be performed by comparing the produced point cloud with another generated by the SfM process. The sparse point cloud should first become dense.

During the process, it was also observed that there were duplicate map points leading to a confusing result. This happens because the system cannot understand that it has passed from this area before and perceives it as a new one. In this way, it reconstructs it again from scratch and due to the drift accumulated over time, the region appears double. However, soon after creation the ORB-SLAM 2 corrected the inconsistency by aligning the two parts of the loop (the duplicated map points were fused), thanks to loop closure thread. Therefore, this procedure is crucial for the process, to achieve optimal results with global geometric consistency. It can be enhanced by the integration of CNNs, both supervised and unsupervised, with the purpose of making it invariant to changing environmental conditions and presence of mobile objects in the scene, which may appear as occlusions. A first idea can be integrated by the work of Memon et al. (2020).

For the evaluation of localization, the examined keyframe trajectory was compared with the groundtruth. Since ORB-SLAM 2 uses an arbitrary coordinate system, the two trajectories had to be aligned first using the Horn’s method. Due to the non-deterministic character of the algorithm, 9 executions were carried out, to extract the median value of the error. The accuracy among the executions was slightly changed, as the ORB-SLAM 2 makes a different initialization each time and therefore, a different initial map is created. The camera poses are computed based on this initial map. The evaluation results show a median translational error of 0.078 m which is competent regarding the conditions of the underwater environment. This

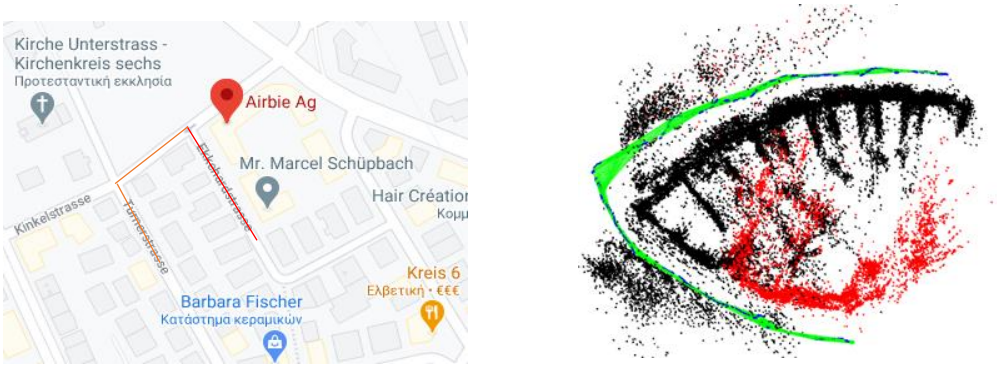
shows that the system can deal with textureless areas and hard conditions. As mentioned in the previous chapter, the accuracy which reflects the cumulative drift in 7 DoFs, depends on many factors such as the surrounding environment and the potentials of the algorithm.
Considering the characteristics of the underwater environment and except from dynamic object removal, a more careful system design as well as ROV's manipulation could be performed, with the aim of reducing the backscattering and turbulence effect caused by the non-perfect placement of strobes and robot's arms, respectively. Although the ORB-SLAM 2 minimizes the error by performing global BA, the proper still occurs due to the pure rotational motions of the camera and large viewpoint changes. When the camera performs pure rotations, the robust triangulation of 3D map points from observations of multiple camera viewpoints is limited, since no sufficient parallax is induced. These map features participate in the trajectory estimation since the system uses them as constraints. This issue can be addressed with the use of omnidirectional cameras such as fisheye lenses. Although the dataset was captured by such a camera, ORB-SLAM 2 provides model only for pinhole cameras and cannot exploit the whole information of these images. It uses planar projection which takes advantage of homogeneous coordinates that are not compatible with fisheye lenses. For this reason, the ORB-SLAM 3 system developed by Mur-Artal et al. (2020) can be leveraged to run the data since it is able to deal with omnidirectional cameras. Additionally, combining the visual sensor with others, like the IMU measurements could significantly boost the robustness and accuracy of the system. The reason behind this, is that the exclusive use of a camera in hard environments like the seabed is not recommended, because the latter is scarce in features. However, the camera is important as the vision is probably the richest source of information for the environment. The examined dataset provided IMU measurements, but it was not possible to leverage this potential within the time frame of this thesis. The extended work on ORB-SLAM 2, i.e., ORB-SLAM 3 enables the integration of IMU measurements into visual SLAM system. Their proposed method estimates the true scale with 5% error in 2 seconds converging to 1% scale error in 15 seconds while simultaneously extracting the direction of gravity which allows to create map aligned with the actual environment.

From the second experiment it is deduced that the algorithm is sensitive to high resolutions, fisheye lenses and the lack of loops during exploration. Keeping the original resolution of the input images (i.e., 1920×1080) the system could process the frames albeit very slowly precluding any chance for real-time process. Thus, it is concluded that the execution time varies highly on the resolution, the under-reconstruction scene and the computational power. After some experiments, it was found that the ORB-SLAM 2 could not handle very resized images since the quality is degraded, and a low number of fps. The results were destructive with the map and trajectory being very confused. Although, the local consistency was achieved to some point, the global consistency was disastrous. The inconsistency is due to the error accumulating over time. This is also caused by the distortion which is very present in images due to the fisheye lens. The images did not get distorted because this is time-consuming. It should be mentioned that the calibration of the camera was performed based on the pinhole camera and not the fisheye. It could be tested using the fisheye model but there would be no significant improvement, since the ORB-SLAM 2 handles only conventional cameras. When the image was resized in such way that contains the 90% of the original, the processing time was increased but a better result arises. The generated map and trajectory roughly followed the geometry of the real environment and the actual path of the MAV, but a curvature in the streets was still introduced. The surfaces, the planes and the relative depth have been well described considering the map result. The system was sensitive to moving objects as mentioned in the previous experiment such as people, birds, and tree leaves. To avoid the erroneous map points and therefore the inaccurate pose estimation, except from the dynamic object removal that was previously mentioned, the enhancement of the map with semantic information could improve the overall accuracy (Yu et al., 2018). The system could rely on these labels and reject points that describe mobile objects. Besides, semantic maps provide conceptual knowledge of the surroundings, boosting the robot's capabilities. When the produced keyframe trajectory was plotted to be compared with the respective groundtruth, the result was different from the one

which displayed during the process. Additionally, the produced trajectory and the reconstructed map were in different scales. The reason behind this needs investigation. Consequently, the quantitative evaluation could not be performed since the error was extremely large.

Figure 57. Difference between the map and trajectory scale.

The process was also tested with more images to see how the error behaves for long-trajectories without the loop closure procedure. Indeed, the accumulated error was getting bigger with the increasing distance traveled, leading to prohibitive results. This reflects the importance of a loop closure inside the monocular system, as it can detect and correct the drift through BA and global optimizations.



(a) (b)

Figure 58. (a) The actual path of the MAV (red line). (b) The estimated trajectory and map from ORB-SLAM 2.

An efficient solution that would have great interest to examine, is the creation of a function that will load an already reconstructed map to the system. Exploiting the localization mode that the ORB-SLAM 2 provides and the save map function that was developed, the system could load the previously built map and localize the position of its current frame within it. In this way, the continuous localization accuracy can be improved, allowing a large-scale reconstruction in an incremental fashion (i.e., create and save the initial map which will be loaded and extended incrementally). According to Mur-Artal et al. (2017), the localization mode assumes that the area is well mapped and no significant changes in the environment occur. During this mode, the mapping and loop closing procedures are deactivated whereas the tracking thread continuously localizes the frames though VO and map point matches. Furthermore, another effort was made, to use another sequence which includes some pure rotational motions. The algorithm was performed well with many matches being detected until some point, where it lost tracking. From there on, the only thing that the algorithm did is trying to relocalize, but due to the lack of visiting previously mapped areas, the relocalization procedure could never be executed. This led to ignoring all the available information present in the rest of the sequence. Restarting the system once the relocalization failed for more than N frames, will increase the number of successfully tracked frames. The algorithm can keep track of submaps, in order not to lose all the previous estimations. Specifically, two threads can run in parallel. The first one could be a procedure that tries to relocalize within the already reconstructed maps and the

second one, will start initializing from scratch and tracking. Once the relocalization finds a match with the known map, the loop closure thread can correct the produced error, to merge the result with the existing maps. The drift will occur since the submaps would have different scale because of the arbitrary initialization. The dataset could also be tested with the ORB-SLAM 3 system (Mur-Artal et al. 2020) since the latter provides a similar approach to submaps.

The main issue that was observed is the drift which accumulates over time and is causing the estimated localization to slowly diverge from its true value. The scale is unknown and hence, the motion estimates and map structure can only be recovered up to scale. For future work, the GPS data (time-synchronized with aerial images) provided by the dataset could be integrated into the algorithm and the results examined, to define the system. Although the GPS measurements are not perfectly accurate due to multipath, atmospheric conditions etc., the system could correct this imprecision via BA and global optimizations. Furthermore, those external data can be integrated into tracking thread and replace the velocity motion model that predicts the next frame's pose, making the relocalization efficient and much faster. A similar solution has been developed by Kiss-Illés et al. (2019). The estimated trajectory can be compared with the GPS positions to evaluate the localization accuracy.
Since the conditions during the elaboration of this diploma thesis did not allow a video capture, this could be done in the future, to evaluate the algorithm online.
Such a system can be used in a variety of applications in robotics and computer vision fields. More specifically, it can be used to maximize the navigation and surveying efforts in difficult environments where the use of GNSS and the human accessibility are unfeasible, to monitor environmental changes and to accomplish surveillance and inspection tasks. Therefore, its further improvements and developments are of great importance.

5.2. Visual Map

To generate an incremental 2D visual map consisting of the keyframes, three similar approaches have been developed. The first effort was computational heavy since it painted each pixel of the final image by searching in which warped image the former had color. To boost its performance as well as make the map incremental, another approach was created. A mask is applied to the warped images and indicates where and where not color exist. The final image is colorized according to these pixels and is visualized "incrementally". For further improvement in terms of performance and time, another function was developed, which also applies a mask to the warped frame and then, calculates the boundaries of the colored image. According to these boundaries, the final image gets colored and can be presented incrementally, since the process is performed for each pair.
The experiments that were conducted were based on the last two approaches. From the results, it was deduced that the proposed methods performed well for a small number of frames (around 15), but as the images were increased, there were misalignments and discontinuities between them. More specifically, the errors existing in the homography estimations are accumulating over time, since the former are transferred to the next homographies due to the consecutive multiplication. This is derived from the fact that the first image in the dataset is used as reference. The errors in the homography reflects the non-planarity of the object. For this reason, the method can be modified in such way to set the middle frame as reference. In this way, the cumulative errors as well as the distortions presented in the aligned images can be reduced. Although, considering the middle image as the reference one makes the integration of the proposed approach into the ORB-SLAM 2 unfeasible, since it could not perform in real-time (alongside the reconstructed 3D map and camera trajectory), it can be an efficient solution for eliminating the errors. Also, due to this multiplication there is a correlation between the estimations, meaning that if a homography is wrongly estimated, this error will be passed to the next ones, corrupting the result. This was also confirmed when the approach was tested for all the keyframes.
Another point that should be examined is the function from which the matches are derived. The other functions the ORB-SLAM 2 uses, can be investigated with the aim of creating a visual

map between frames, not necessarily keyframes with a specific step. Therefore, the error produced will not accumulate at the same rate as it does now. The integration of such a visual map inside the algorithm is important as it can be used to enhance the navigation, i.e., the user can be informed if he had passed through an area again.
With respect to the execution time of both approaches, it was observed that the second method was much faster than the first one (for the reason described above). Also, the growth rate of the time was much lower in the second case, since the first one is more sensitive to the large number of frames. For this reason, the images were resized so that the width and height being the 60% of the original. The time was decreased, and the output images were well-aligned.
Furthermore, during the alignment, it was noticed that the seam between the keyframes was present. The seams are presented as discontinuities and stitching lines and hence, they can be detected and eradicated making the overlap area smoother and more natural.
Additionally, when a big number of images is used for the creation of the visual map, many of them are cropped since the dimensions of the final "mosaic" are not sufficient. The development of a graphical user interface (GUI) incorporated into the ORB-SLAM 2 system could be an efficient solution, so that the boundaries of the final image will change automatically.
It should also be pointed out that in the generated "homography.txt" file the first correspondences are stored in reverse leading to confusion when the program reads them. However, it was not possible to investigate it within the time frame of this diploma thesis.

Appendix

1. Motion-only Bundle Adjustment: is used in tracking thread to optimize the camera pose (orientation R and position t) of the processed keyframe whereas map points remain fixed. It is based on minimizing the reprojection error between matched 3D points X^i (in world coordinate system) and keypoints x^i (monocular x^{im} or stereo x^{is}), where i belongs to χ, the set of all correspondences. It can be described as follows:

$$(R,t) = argmin_{R,t} \sum \rho * (\left|\left| x^i - \pi * (R * X^i + t \right|\right|_{\Sigma}^2$$

, where ρ is the Huber cost function and Σ the covariance matrix associated to the scale at which the keypoint was detected. π is the projection function which is defined below either for monocular or stereo points:

$$\pi_m = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x * \frac{X}{Z} + c_x \\ f_y * \frac{Y}{Z} + c_y \end{bmatrix}$$

$$\pi_s = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x * \frac{X}{Z} + c_x \\ f_y * \frac{Y}{Z} + c_y \\ f_x * \frac{X - b}{Z} + c_x \end{bmatrix}$$

, where (f_x, f_y) is the focal length, (c_x, c_y) is the principal point and b, the baseline. They are all known from camera calibration.

2. Local Bundle Adjustment: is used for optimizing a set of covisible keyframes (K_1 and K_2) in the covisibility graph and map points P_L,, observed in these keyframes. All other keyframes, K_F that see the same map points P_L,, participate in the optimization but remain fixed. The optimization problem is the following:

$$X^i, R_l, t_l \left| \, i \in P_L, l \in K_1 \& K_2 \right\} = \; argmin_{X^i, R_l, t_l} \sum_{k \epsilon K_1, K_2} \sum_{j \epsilon X^k} \rho(E_{kj})$$

$$E_{kj} = \; \left|\left| x_j - \pi * (R_k * X^j + t_k) \right|\right|_{\Sigma}^2$$

, where X^k is the set of matches between points P_L and keypoints in a keyframe k.

3. Full Bundle Adjustment: is a special case of local BA where all map points and keyframes are participated except for the first keyframe which remain fixed as the origin (for gauge freedom elimination).

4. Perspective n Point problem (PnP): is a problem of estimating the pose of a camera (6 DoF-translation and rotation), assuming that it is calibrated, given a set of 3D to 2D correspondences, i.e., 3D points (p_w) in the world coordinate system and their 2D corresponding projections in the image (p_c). It can be described as follows:

$$s * p_c = K[R|T] * p_w = s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

, where s is a scale factor of a point in the image, $p_c$is the homogeneous corresponding image point, K is the matrix of camera's intrinsic parameters, R, T is the 3D rotation and 3D translation respectively and p_w is the homogeneous world point.
Specifically, $(f_x,\ f_y)$ is the scaled focal length, γ is the skew parameter (in most cases equals to 0) and (u_0, v_0) is the principal point.
Some methods such as UPnP and DLT do not need a calibrated camera in the beginning, thus both intrinsic and extrinsic parameters will be calculated. The two most common methods that are used for solving the PnP problem are the P3P method and EPnP (Efficient PnP) with the latter, being used in ORB-SLAM 2 implementation. More precisely, EPnP is a method developed by Lepetit et al., (2008), which solves the above problem using n>=4 correspondences. It is based on the notion that that each of the n points (reference points) can be expressed as a weighted sum of 4 virtual control points (PnP, Wikipedia). Hence, the unknowns of the system are the coordinates of those points which will later lead on the camera pose estimation. Each world point p_{iw} and its corresponding image point p_{ic} can be described as weighted sums of 4 control points, c_{iw} and c_{ic} respectively as:

$$p_{iw} = \sum_{(j=1,4)} a_{ij} * c_{iw}$$

$$p_{ic} = \sum_{(j=1,4)} a_{ij} * c_{ic}$$

The weights are normalized per reference point as: $\sum_{(j=1,4)} a_{ij} = 1$. Thus, the equation for the system becomes:

$$s_i * p_{ic} = K * \sum_{j=1,4)} a_{ij} * c_{ic} \ (1)$$

where $c_{ic} = [x_{ic} \quad y_{ic} \quad z_{ic}]^T$ is the homogeneous coordinates of an image control point. Each reference point is described by the following two linear equations, produced by (1):

a. $\sum_{(j=1,4)} a_{ij} * f_x * x_{ic} + a_{ij} * (u_0 - u_1) * z_{ic} = 0$
b. $\sum_{(j=1,4)} a_{ij} * f_y * y_{ic} + a_{ij} * (v_0 - v_1) * z_{ic} = 0$

The system can be formed as: $Mx=0$ where $x = [c_{1c}^T \quad c_{2c}^T \quad c_{3c}^T \quad c_{4c}^T]^T$ and the solution of control points is given:

$$x = \sum_{(i=1,N)} \beta_i * v_i$$

, where N is the number of null singular values in M (and can range from 0 to 4), β_i are the coefficients and $v_i$is the corresponding right singular vector of M. The initial coefficients are refined using the Gauss-Newton algorithm (which solves non-linear least squares problems without computing the second derivatives). Then, the R and T matrices are computed minimizing the reprojection error. The advantage is that the complexity of the solution increases linearly an in direct proportion to the inputs and can effectively work for both planar and non-planar surfaces. Also, due to outlier existing in the set of correspondences, the RANSAC algorithm is used in collaboration with the EPnP method in order to make the final pose estimation more robust.

References

Amiri J. and Moradi H. (2016) 'Real-time video stabilization and mosaicking for monitoring and surveillance', 4th International Conference on Robotics and Mechatronics (ICROM), pp. 613-618. IEEE.

Association for Advancing Automation (2018), 'Visual SLAM technology benefits and applications' [Online]. Available at: https://www.controleng.com/articles/visual-slam-technology-benefits-and-applications/ (Accessed: 5 January 2021).

'Backscattering' (2020). *Wikipedia* [Online]. Available at: https://en.wikipedia.org/wiki/Backscatter (Accessed: 6 March 2021).

Balntas, V., Riba, E., Ponsa, D. and Mikolajczyk, K. (2016) 'Learning local feature descriptors with triplets and shallow convolutional neural networks', *Proceedings of the British Machine Vision Conference,* pp.119.1-119.11.

Bay H., Tuytelaars T. and Van Gool L. (2006) 'Surf: Speeded up robust features', *European Conference on Computer Vision*, pp. 404-417.

Cadena C., Carlone L., Carrillo H., Latif Y., Scaramuzza D., Neira J. and Leonard J. J. (2016) 'Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age', *IEEE Transactions on Robotics*, 32(6), pp. 1309-1332.

Calonder M., Lepetit V., Strecha C. and Fua P. (2010) 'Brief: Binary robust independent elementary features', *European Conference on Computer Vision*, pp. 778-792.

'Capo Sagro 3' (2021) [Online]. Available at: https://archeologie.culture.fr/archeo-sous-marine/en/capo-sagro-3-coast-bastia-corsica (Accessed: 5 March 2021).

Chotrov D., Yordanov Y., Uzunova Z. and Maleshkov S. (2018) 'Mixed-Reality Spatial Configuration with a ZED Mini Stereoscopic Camera', *Technologies and Education for a Smart World.*

Civera J., Grasa O. G. Davison A. J. and Montiel, J. M. M. (2009) '1-point RANSAC for EKF-based structure from motion', *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3498-3504.

'CloudCompare' (2021) [Online]. Available at: https://www.danielgm.net/cc/ (Accessed: 28 February 2021).

Cremers D., Schöps T. and Engel J. (2020) 'LSD-SLAM: Large-Scale Direct Monocular SLAM', [online] *Vision In Tum* [Online]. Available at: https://vision.in.tum.de/research/vslam/lsdslam *(*Accessed: 19 December 2020).

Davison A. J., Reid I. D., Molton N. D. and Stasse O. (2007) 'MonoSLAM: Real-time single camera SLAM', *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6),* pp. 1052-1067.

Durrant-Whyte H. and Bailey T. (2006) 'Simultaneous localization and mapping: part I.', *IEEE Robotics & Automation Magazine*, 13(2), pp. 99-110.

Dusmez S., Heydarzadeh M., Nourani M. and Akin B. (2017) 'Remaining useful lifetime estimation for power mosfets under thermal stress with RANSAC outlier removal', *IEEE Transactions on Industrial Informatics*, 13(3), pp. 1271-1279.

'Eigen' (2021) [Online]. Available at: https://eigen.tuxfamily.org/index.php?title=Main_Page (Accessed: 28 February 2021).

Einicke G. and White L. (1999) 'Robust extended Kalman filtering', *IEEE Transactions on Signal Processing*, 47(9), pp. 2596-2599.

Engel J., Schöps T. and Cremers D. (2014) 'LSD-SLAM: Large-scale direct monocular SLAM', In *European Conference on Computer Vision*, pp. 834-849, Springer, Cham.

'Epipolar geometry' (2020), *Wikipedia* [Online]. Available at: https://en.wikipedia.org/wiki/Epipolar_geometry (Accessed: 22 December 2020).

Fischler M and Bolles R. (1981) 'Random sample consensus: A paradigm for modelfitting with applications to image analysis and automated cartography', *Commu-nications Of The ACM*, 24(6), pp. 381–395.

Ferrera M., Creuze V., Moras J. and Trouvé-Peloux P. (2019) 'AQUALOC: An underwater dataset for visual–inertial–pressure localization', *The International Journal of Robotics Research*, *38*(14), pp. 1549-1559.

'*Fotokite*' [Online]. Available at: https://fotokite.com/ [Accessed: 20 February 2021].

Galvez-López D. and Tardos J. D. (2012) 'Bags of Binary Words for Fast Place Recognition in Image Sequences', IEEE Transactions On Robotics, 28(5), pp. 1188-1197, doi: 10.1109/TRO.2012.2197158.

Handa A., Whelan T., McDonald J. and Davison A. (2014) 'A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM', IEEE International Conference on Robotics and Automation, pp. 1524-1531.

Harris G. and Stephens M. (1988) 'A combined corner and edge detector', *Alvey Vision Conference*, 15(50), pp. 10-5244.

Hartley R. and Zisserman A. (2003) '*Multiple view geometry in computer vision',* Cambridge university press.

Jiang S., Jiang C., & Jiang W. (2020) 'Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools', *ISPRS Journal of Photogrammetry and Remote Sensing*, *167*, pp. 230-251.

Kang R., Shi J., Li X., Liu Y. and Liu X. (2019) 'DF-SLAM: A deep-learning enhanced visual SLAM system based on deep local features', *arXiv preprint arXiv:1901.07223*.

Karmacharya K., Bishwakarma M., Shrestha U. and Rüther N. (2019) 'Application of 'Structure from Motion'(SfM) technique in physical hydraulic modelling', *Journal of Physics: Conference Series*, 1266(1), pp. 012008.

Klein G. and Murray D. (2007) 'Parallel tracking and mapping for small AR workspaces', *6th IEEE and ACM International Symposium on Mixed and Augmented* Reality, pp. 225-234.

Kora K., Jonnalagadda S. (2016) ‘Image Mosaicing using Feature based technique’, *International Journal of Scientific & Engineering Research*, 7(6), pp. 1117-1125.

Kourounioti O. (2018) ‘Large scale orthophoto production using SLAM technique – implementation in Kalamoti of Chios island’ Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (https://dspace.lib.ntua.gr/xmlui/handle/123456789/47035) (in Greek).

Kümmerle R., Grisetti G., Strasdat H., Konolige K. and Burgard, W. (2011) ‘g2o: A general framework for graph optimization’, *IEEE International Conference on Robotics and Automation*, pp. 3607-3613.

Levenberg K. (1944) ‘A method for the solution of certain nonlinear problems’, *Q. Appl. Math.*, *2*, pp. 164-168.

Liu S., Guo P., Feng L. and Yang A. (2019) ‘Accurate and Robust Monocular SLAM with Omnidirectional Cameras’, *Sensors*, 19(20), pp. 4494.

Lowe G. (1999) ‘Object recognition from local scale-invariant features’, *Proceedings of the Seventh IEEE International Conference on Computer Vision,* 2, pp. 1150-1157.

Majdik L., Till C. and Scaramuzza D. (2017) ‘The Zurich urban micro aerial vehicle dataset’, *The International Journal of Robotics Research*, 36(3), pp. 269-273.

Martínez-Carranza J., Loewen N., Márquez F., García E. and Mayol-Cuevas W. (2015) ‘Towards autonomous flight of micro aerial vehicles using ORB-SLAM’, *IEEE Workshop on Research, Education and Development of Unmanned Aerial* Systems, pp. 241-248.

Mishchuk A., Mishkin D., Radenovic F. and Matas J. (2017) ‘Working hard to know your neighbor's margins: Local descriptor learning loss’, In *Advances in Neural Information Processing Systems*, pp. 4826-4837.

Mur-Artal R., Montiel J. M. M. and Tardos J. D. (2015) ‘ORB-SLAM: a versatile and accurate monocular SLAM system’, *IEEE Transactions on Robotics*, *31*(5), pp. 1147-1163.

Mur-Artal R. and Tardós J. D. (2017) ‘Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras’, *IEEE Transactions on Robotics*, *33*(5), pp. 1255-1262.

Mur-Artal R. and Tardós, J. D. (2014) ‘Fast relocalisation and loop closing in keyframe-based SLAM’, In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 846-853.

Mur-Artal R. and Tardós J. D. (2014) ‘Orb-slam: Tracking and mapping recognizable’, *Work. Multi View Geom. Robot.*

Newcombe R. A. Lovegrove S. J. and Davison A. J. (2011) ‘DTAM: Dense tracking and mapping in real-time’, I*nternational Conference on Computer Vision*, pp. 2320-2327.

Nistér D. (2004) ‘An efficient solution to the five-point relative pose problem’, *Transactions on Pattern Analysis and Machine Intelligence*, 26(6), pp. 756-770.

Nistér D., Naroditsky O. and Bergen, J. (2004) ‘Visual odometry’, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (1), pp. I-I.

Nocerino E., Menna F., Chemisky B. and Drap P. (2020) '3D sequential image mosaicing for underwater navigation and mapping', *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43, pp. 991-998.

'OpenCV' (2021) [Online]. Available at: opencv.org (Accessed 28 February 2021).

Ortiz-Coder P. and Sánchez-Ríos A. (2019) 'A Self-Assembly Portable Mobile Mapping System for Archeological Reconstruction Based on VSLAM-Photogrammetric Algorithm', *Sensors*, 19(18), pp. 3952.

Ozyesil O., Voroninski V., Basri R. and Singer, A. (2017) 'A survey of structure from motion', *arXiv preprint arXiv:1701.08493*.

'Pix4D Documentation, Yaw, Pitch and Roll angles' [Online]. Available at: https://support.pix4d.com/hc/en-us/articles/202558969-Yaw-Pitch-Roll-and-Omega-Phi-Kappa-angles (Accessed: 5 February 2021).

'Point Cloud Data' (2021) [Online]. Available at: https://pointclouds.org/documentation/tutorials/pcd_file_format.html (Accessed 6 March 2021).

Pulcrano M., Scandurra S., Minin G. and Luggo A. (2019) '3D CAMERAS ACQUISITIONS FOR THE DOCUMENTATION OF CULTURAL HERITAGE', *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Ramirez E. (2017) 'What is meant by cultural heritage'. UNESCO [Online]. Available at: http://www.unesco.org/new/en/culture/themes/illicit-trafficking-of-cultural-property/unesco-database-of-national-cultural-heritage-laws/frequently-asked-questions/definition-of-the-cultural-heritage/ (Accessed: 5 January 2021).

'Random sample consensus' (2020), *Wikipedia* [Online]. Available at: https://en.wikipedia.org/wiki/Random_sample_consensus (Accessed: 22 December 2020).

Rosten E. and Drummond T. (2006) 'Machine learning for high-speed corner detection', *European Conference on Computer Vision*, pp. 430-443.

Rublee E., Rabaud V., Konolige K. and Bradski, G. (2011) 'ORB: An efficient alternative to SIFT or SURF', *International Conference on Computer Vision*, pp. 2564-2571.

Salas M., Latif Y., Reid D. and Montiel M. (2015) 'Trajectory alignment and evaluation in SLAM: Horns method vs alignment on the manifold', *Robotics: Science and Systems Workshop: The problem of mobile sensors*.

Scaramuzza D. and Fraundorfer F. (2011) 'Visual odometry [tutorial]', *IEEE robotics & automation magazine, 18*(4), pp. 80-92.

Schonberger L. and Frahm M. (2016) 'Structure-from-motion revisited', In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104-4113.

Schönberger L., Fraundorfer F. and Frahm M. (2014) 'STRUCTURE-FROM-MOTION FOR MAV IMAGE SEQUENCE ANALYSIS WITH PHOTOGRAMMETRIC APPLICATIONS', *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Sheng L., Xu D., Ouyang W. and Wang, X. (2019) 'Unsupervised collaborative learning of keyframe detection and visual odometry towards monocular deep SLAM', In *Proceedings of the IEEE International Conference on Computer Vision,* pp. 4302-4311.

Singandhupe A. and La H. (2019) 'A review of slam techniques and security in autonomous driving', *Third IEEE International Conference on Robotic Computing (IRC),* pp. 602-607

Strasdat, H., Montiel, J. M. and Davison, A. J. (2012) 'Visual SLAM: why filter?', *Image and Vision Computing, 30*(2), pp. 65-77.

Sturm J., Burgard W. and Cremers D. (2012) 'Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark', *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*.

Sumikura S., Shibuya M. and Sakurada K. (2019) 'OpenVSLAM: a versatile visual SLAM framework', In *Proceedings of the 27th ACM International Conference on Multimedia,* pp. 2292-2295.

Taketomi T., Uchiyama H. and Ikeda S. (2017) 'Visual SLAM algorithms: a survey from 2010 to 2016', *IPSJ Transactions on Computer Vision and Applications, 9*(1), 16.

Thoeni K., Irschara A. and Giacomini, A. (2012) 'Efficient photogrammetric reconstruction of highwalls in open pit coal mines', In *16th Australasian Remote Sensing and Photogrammetry Conference,* pp. 85-90.

'Turbidity' (2020). *Wikipedia* [Online]. Available at: https://en.wikipedia.org/wiki/Turbidity (Accessed: 6 March 2021).

Wang S., Clark R., Wen H. and Trigoni N. (2017) 'DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks', *IEEE International Conference on Robotics and Automation (ICRA),* pp.2043-2050, doi:10.1109/ICRA.2017.7989236.

'Wikitude SLAM technology' [Online]. Available at: https://www.wikitude.com/wikitude-slam/ (Accessed: 5 January 2021).

Wu C. (2013) 'Towards linear-time incremental structure from motion', *International Conference on 3D Vision-3DV,* pp. 127-134.

Yousif K., Bab-Hadiashar A. and Hoseinnezhad R. (2015) 'An overview to visual odometry and visual SLAM: Applications to mobile robotics', *Intelligent Industrial Systems,* 1(4), pp. 289-311.

Zhang, Z. (2000) 'A flexible new technique for camera calibration', *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), pp. 1330-1334.

Zhang Z. and Scaramuzza D. (2018) 'A tutorial on quantitative trajectory evaluation for visual (inertial) odometry', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*
pp. 7244-7251.