ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ & ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

ΕΡΓΑΣΤΗΡΙΟ ΦΩΤΟΓΡΑΜΜΕΤΡΙΑΣ

Αυτοματοποιημένος Εντοπισμός Ακμών σε Νέφη Σημείων με τη Βοήθεια Σημασιολογικής Πληροφορίας

Automated Detection of Edges in Point Clouds using Semantic Information

Διπλωματική Εργασία

Θοδωρής Μπέτσας

Επιβλέπων:
Ανδρέας Γεωργόπουλος
Καθηγητής ΕΜΠ

Αθήνα Μάρτιος 2021

στο Βασίλη, την Αιμιλία

και την Αγγελική

Ευχαριστίες

Με το πέρας της διπλωματικής μου εργσίας (ΔΕ) θα ήθελα να ευχαριστήσω τόσο όσους συνέβαλαν σε αυτή αλλά και όσους συνέβαλαν στην ολοκλήρωση των σπουδών μου εν γένει.

Αρχικά θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Ανδρέα Γεωργόπουλο, για την αμέριστη βοήθεια του σε αυτή τη διπλωματική εργασία αλλά και ως Δάσκαλο, που από τα προηγούμενα έτη με βοήθησε να κατανοήσω αλλά και να αγαπήσω το αντικείμενο του Αγρονόμου και Τοπογράφου Μηχανικού (ΑΤΜ) και ειδικά της Φωτογραμμετρίας.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Χρήστο Ιωσηφίδη, μέλος ΕΕΔΙΠ που «χαιρετήσαμε το κόσμο» του προγραμματισμού αλλά και εκείνο του ΑΤΜ και με βοήθησε να αναπτύξω νέες δεξιότητες από τα πρώτα έτη των σπουδών μου.

Επιπλέον θα ήθελα να ευχαριστήσω την υποψήφια διδάκτορα κα. Έλλη Σταθοπούλου για τις ουσιαστικές παρατηρήσεις της στη διάρκεια αυτής της ΔΕ.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου, που είναι δίπλα μου σε ό,τι και αν προκύψει και με στηρίζουν σε κάθε μου βήμα.

Περίληψη

Στην παρούσα διπλωματική εργασία δημιουργήθηκε ένας αλγόριθμος εξαγωγής 3Δ ακμών από νέφη σημείων χρησιμοποιώντας ψηφιακές εικόνες και σημασιολογική πληροφορία. Ο αλγόριθμος αναπτύχθηκε σε γλώσσα Python και διανέμεται στο GitHub (www.github.com/thobet) ενώ μπορεί να χρησιμοποιηθεί με οποιοδήποτε λειτουργικό σύστημα.

Αρχικά διερευνήθηκε η μεταφορά της σημασιολογικής πληροφορίας στο νέφος σημείων μέσω των ψηφιακών εικόνων. Για το σκοπό αυτό οι εικόνες εμπλουτίζονται με ένα τέταρτο κανάλι που περιέχει το χάρτη ακμών τους. Στη συνέχεια εισάγονται σε ένα λογισμικό δομής από κίνηση (SfM) και πυκνής συνταύτισης (MVS) ώστε να παραχθεί αρχικά το αραιό και εν τέλει το πυκνό νέφος σημείων. Για τη μελέτη του συγκεκριμένου βήματος αρχικά δημιουργήθηκε ένα πρόγραμμα τριγωνισμού στο οποίο εισάγονται οι εμπλουτισμένες εικόνες και παράγεται ένα αραιό νέφος σημείων. Τα σημεία του αραιού νέφους χαρακτηρίζονται από επτά τιμές. Τρεις για τη θέση, τρείς για το χρώμα και μια για το αν το εκάστοτε σημείο ανήκει σε ακμή ή όχι. Στη συνέχεια τροποποιήθηκε το λογισμικό OpenSfM [OpenSfM] ώστε να δέχεται εικόνες τεσσάρων καναλιών και να μεταφέρει τη σημασιολογική πληροφορία στο πυκνό νέφος σημείων που παράγει. Επιπλέον χρησιμοποιήθηκε και το λογισμικό Agisoft Metashape [Agisoft-Metashape] το οποίο μπορεί και επεξεργάζεται πολυκάναλες εικόνες ενώ μεταφέρει τις τιμές κάθε καναλιού στα σημεία τόσο του αραιού όσο και του πυκνού νέφους. Στην παρούσα διπλωματική εργασία αναπτύχθηκε και ένα πρόγραμμα το οποίο εκτελεί τους αλγορίθμους SfM-MVS του Agisoft Metashape [Agisoft-Metashape] μέσω της βιβλιοθήκης Python που παρέχει. Έτσι δίνεται η δυνατότητα να χρησιμοποιηθεί τόσο το γραφικό περιβάλλον του Metashape [Agisoft-Metashape] όσο και να εκτελεστούν οι αλγόριθμοι απευθείας από την Python. Ο χρήστης επιλέγει μια από τις προσεγγίσεις, (i) τριγωνισμός, (ii) OpenSfM [OpenSfM] και (iii) Agisoft Metashape GUI [Agisoft-Metashape] ή script, ώστε να παράξει το αραιό ή το πυκνό νέφος που περιέχει την πληροφορία των ακμών. Ο χρήστης εκτός της μεθόδου που θα παραχθεί το αραιό ή το πυκνό νέφος επιλέγει και τον τρόπο που θα οριστεί ο χάρτης ακμών δηλαδή η σημασιολογική πληροφορία. Πιο συγκεκριμένα μπορεί είτε να δώσει την πληροφρία ο ίδιος, στην περίπτωση που έχει υπολογιστεί ήδη ο χάρτης ακμών κάθε εικόνας, είτε να τη δημιουργήσει εφαρμόζοντας σε πραγαμτικό χρόνο [Arapellis (2020)] τον αλγόριθμο Canny [Canny (1983) και (1986)] σε κάθε μια. Στη συνέχεια εξάγονται τα σημεία που ανήκουν σε ακμές και ταξινομούνται σε σημεία της κάθε ακμής. Τέλος κάθε ομάδα σημείων διανυσματοποιείται και έτσι παράγεται ένα προσεγγιστικό 3Δ σχέδιο του αντικειμένου μελέτης.

Ο αλγόριθμος εφαρμόστηκε σε δύο μνημεία πολιτιστικής κληρονομιάς, στο Ναό της Θεάς Δήμητρας στη Νάξο και στον Παλιό Αστυνομικό Σταθμό στη Ρόδο. Στα συμπεράσματα μελετάται η αποτελεσματικότητα του αλγορίθμου ενώ παρουσιάζονται και βελτιώσεις που μπορούν να προστεθούν στο μέλλον.

Abstract

In this diploma thesis, an algorithm for extracting 3D edges from point clouds was developed, combining digital images with semantic information. The proposed algorithm was built in Python and is distributed in GitHub (www.github.com/thobet). It can be used by any operating system.

At first, the transition of semantic information to point clouds through digital images is investigated. For this purpose, the images are enriched by a fourth channel which contains their edge map. Afterwards, they are inserted into a SfM-MVS software to produce initially a sparse and finally, a dense point cloud. For the investigation of this procedure, a triangulation program was first created. More precisely, the enriched images are inserted, and the sparse point cloud is generated. Each of its points is defined by 7 values, i.e. three for the position, three for the color and one which determine whether the point belongs to an edge or not. The OpenSfM [OpenSfM] software was modified in such way, to accept four-channel images and transfer the semantic information to the produced dense point cloud. This transition can also be made using the Agisoft Metashape [Agisoft-Metashape] software which can process multi-channel images by default. Apart from that, it can transfer the semantic information to sparse point cloud, too. This diploma thesis also proposes a program which executes the SfM-MVS algorithms of Agisoft Metashape [Agisoft-Metashape] through the library that it provides. In this way, it enables one to use both the Metashape [Agisoft-Metashape] graphical interface and run the algorithms directly from Python. The user can choose one of the proposed approaches, i.e. triangulation, OpenSfM [OpenSfM] and Agisoft Metashape [Agisoft-Metashape] GUI or script, to produce the sparse or dense point cloud that contains the edge information. He can also decide about the way the edge map will be defined, i.e. the semantic information. Specifically, he can either provide the information himself in case the edge map has already been computed for each image or create it, by applying the Canny [Canny (1983) and (1986)] algorithm to each image, in real-time [Arapellis (2020)]. The points belonging to edges are extracted and classified into points of each edge. Finally, each segment of points is vectorized and thus, the approximated 3D plan of the object of interest is produced.

The proposed algorithm was applied to two monuments of cultural heritage, the Ancient Temple of Demeter in Naxos, and the old Police Station in Rhodes. In the conclusions, the effectiveness of the algorithm is investigated while presenting improvements that can been added in the future.

Table of Contents

0. Introduction - Scope of Study 18
1. State of the art 21
1.1. Direct Methods 21
1.2. Indirect Methods: 24
1.3. Commercial Software 29
2. Theoretical Approach 32
2.1 Structure from Motion – Multi View Stereo 32
2.2 Semantic Information 47
3. Practical Implementation 59
3.1 Temple of Demeter, Naxos Dataset 59
3.2 Old Police Station, Monolithos, Rhodes Dataset 60
3.3 Structure from Motion - Multi View Stereo and other Software 62
3.3.2 Mapillary - OpenSfM 62
3.3.3 AliceVision – Meshroom 63
3.3.4 CloudCompare 63
3.4 Developed Approaches 63
3.4.1 Implementation using the MyTriangulation pipeline 64
3.4.2 Implementation using the OpenSfM software 70
3.4.3 Implementation using the Metashape software 78
3.5 Proposed Approach 86
4. Conclusions 102

Εισαγωγή

Η τεχνολογική εξέλιξη, τόσο της επιστήμης της φωτογραμμετρίας, όσο και της όρασης υπολογιστών, λόγω της ραγδαίας ανάπτυξης των ηλεκτρονικών υπολογιστών, έχει οδηγήσει στη λύση πληθώρας πολύπλοκων προβλημάτων. Φωτογραμμετρία καλείται η επιστήμη που ασχολείται με την εξαγωγή αξιόπιστης τρισδιάστατης γεωμετρικής πληροφορίας αντικειμένων, χρησιμοποιώντας εικόνες ή συλλέγοντας αποστάσεις εκμεταλλευόμενη διάφορα φαινόμενα της ηλεκτρομαγνητικής ακτινοβολίας. Αντίστοιχα Όραση Υπολογιστών καλείται η επιστήμη που μελετά μεθόδους, οι οποίες καθιστούν τον ηλεκτρονικό υπολογιστή ικανό να “αντιληφθεί” το περιβάλλον στο οποίο βρίσκεται, χρησιμοποιώντας ως δεδομένα εικόνες, βίντεο ή διάφορες μεθόδους μέτρησης αποστάσεων. Στην πραγματικότητα, μελετάται πώς ένας ηλεκτρονικός υπολογιστής μπορεί να αντεπεξέλθει σε προβλήματα τα οποία ο ανθρώπινος εγκέφαλος εκτελεί με παραδειγματική ευκολία, όπως η αναγνώριση αντικειμένων ή χρωμάτων κλπ. Ευρέως χρησιμοποιούμενα φωτογραμμετρικά παράγωγα αποτελούν οι ορθοφωτογραφίες, οι ορθοφωτοχάρτες, τα τρισδιάστατα μοντέλα καθώς και δισδιάστατα – τρισδιάστατα αρχιτεκτονικά σχεδία. Τα 2Δ – 3Δ αρχιτεκτονικά σχέδια αποτελούν το πιο διαδεδομένο τρόπο αναπαράστασης ενός αντικειμένου, με ένα ευρέως κατανοητό τρόπο. Ωστόσο, η παραγωγή τους σε 2Δ πολύ δε περισσότερο σε 3Δ, είναι μια χειροκίνητη, δύσκολη και επίπονη διαδικασία, ακόμα και σήμερα. Λόγω των παραπάνω και σε συνδυασμό με το ότι τα αρχιτεκτονικά σχέδια δε θα σταματήσουν να αποτελούν ένα από τα κύρια παράγωγα τόσο για τους μηχανικούς όσο και για πολλούς άλλους επιστημονικούς κλάδους, η βελτίωση και εν τέλει η αυτοματοποίηση της διαδικασίας παραγωγής τους καθίσταται αναγκαία. Συχνά οι επιστήμες της φωτογραμμετρίας και της όρασης υπολογιστών χρησιμοποιούνται για την μελέτη των μνημείων πολιτιστικής κληρονομιάς. Ως πολιτιστική κληρονομιά ορίζονται τα φυσικά αντικείμενα καθώς και τα άυλα χαρακτηριστικά, μιας ομάδας ή κοινωνίας τα οποία κληροδοτούνται στις νέες γενιές. Μια κοινωνία ή μια ομάδα ορίζει την πολιτιστική της κληρονομιά ενώ παράλληλα αυτή αποτελεί μέρος της παγκόσμιας πολιτιστικής κληρονομιάς. Κάθε γενιά οφείλει να προστατεύσει και να κληροδοτήσει στη νέα, τόσο την άυλη όσο και την υλική πολιτιστική κληρονομιά. Ο σκοπός της παρούσας διπλωματικής εργασίας είναι ο εντοπισμός και η εξαγωγή τρισδιάστατων ακμών από νέφη σημείων χρησιμοποιώντας σημαντική-σημασιολογική πληροφορία και ψηφιακές εικόνες. Ακόμα, ερευνά την δυνατότητα αυτόματης διανυσματοποίησης των εξαγόμενων ακμών και εν τέλει τη δημιουργία ενός όσο το δυνατόν πληρέστερου τρισδιάστατου σχεδίου. Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκαν εικόνες από δύο μνημεία, το Ναό της Θεάς Δήμητρας στη Νάξο και τον Παλιό Αστυνομικό Σταθμό στον Μονόλιθο της Ρόδου, από τις οποίες παρήχθησαν τα πυκνά νέφη σημείων. Αυτά εκτός της θέσης και του χρώματος κάθε σημείου, φέρουν και μια τιμή “δείκτη” (label) η οποία υποδηλώνει αν το σημείο ανήκει σε ακμή ή όχι.

Αρχικά παρουσιάζονται κάποιες μέθοδοι, που έχουν ήδη πραγματοποιηθεί και ανήκουν στην ευρύτερη θεματολογία της παρούσας διπλωματικής εργασίας. Στην συνέχεια παρουσιάζονται τρεις προσεγγίσεις που αναπτύχθηκαν για τη διερεύνηση της εξαγωγής ακμών από νέφος σημείων χρησιμοποιώντας σημασιολογική πληροφορία. Έπειτα, παρουσιάζεται ο προτεινόμενος αλγόριθμος, καθώς και δύο εφαρμογές του. Μια στον Αρχαίο Ναό της Θεάς Δήμητρας στην Νάξο και μια στον Παλιό Αστυνομικό Σταθμό, του χωριού Μονόλιθου στη Ρόδο. Τέλος, πραγματοποιήθηκε μια σύγκριση μεταξύ εφαρμογών με διαφορετική παραμετροποίηση, μια ευρύτερη αξιολόγηση της προτεινόμενης μεθόδου καθώς και βελτιώσεις που μπορούν να διερευνηθούν και να συμπεριληφθούν στο μέλλον.

Αντίστοιχες Προσπάθειες

Οι τεχνικές εξαγωγής ακμών χρησιμοποιούνται για πληθώρα εφαρμογών, όπως η αναγνώριση των λωρίδων κυκλοφορίας, η αναγνώριση αντικειμένων κ.ά. Γενικότερα, οι μέθοδοι εξαγωγής τρισδιάστατων ακμών ταξινομούνται σε δύο ευρύτερες κατηγορίες, τις άμεσες και τις έμμεσες. Άμεσες καλούνται οι μέθοδοι που εφαρμόζονται απευθείας σε μετρητικά δεδομένα, για παράδειγμα σε ένα νέφος σημείων, δηλαδή σε περιπτώσεις που δεν είναι διαθέσιμες εικόνες. Σε αντίθεση με τις άμεσες, οι έμμεσες

αποσκοπούν στην εξαγωγή τρισδιάστατων ακμών, εντοπίζοντάς τες πρώτα σε εικόνες και έπειτα στον 3Δ χώρο. Στην συνέχεια παρουσιάζονται διάφορες, τόσο άμεσες όσο και έμμεσες, προσεγγίσεις.

Άμεσες Μέθοδοι

Η Mitropoulou (2017) και οι Mitropoulou & Georgopoulos (2019) παρουσίασαν μια μέθοδο κατά την οποία ο εντοπισμός των τρισδιάστατων ακμών, πραγματοποιείται μέσω της τομής δύο επιπέδων. Η μέθοδος χρησιμοποιεί ως δεδομένα εισόδου νέφη σημείων. Στη συνέχεια προσδιορίζει τα δύο επίπεδα και εν τέλει εξάγει την 3Δ ακμή που ορίζεται από την τομή των δύο επιπέδων. Ο αλγόριθμος που δημιουργήθηκε εφαρμόστηκε σε δεδομένα από το ναό του Ηφαίστου, στην αρχαία αγορά της Αθήνας.

Ο Bienert (2008) παρουσίασε μια μέθοδο διανυσματοποίησης και διαστασιολόγησης ακμών που έχουν εξαχθεί από φιλτραρισμένα νέφη σημείων, που δημιουργούνται από σαρωτές λέιζερ. Αρχικά εξάγονται διάφορες τομές – οριζόντιες και κάθετες - από το νέφος σημείων τέμνοντάς το με επίπεδα, παράλληλα στον X ή Y άξονα καθώς και συγκριμένου πάχους. Τα εξαγόμενα σημεία φιλτράρονται μέσω της προσαρμογής μιας γραμμής ή ενός τόξου σε αυτά, για να απομακρυνθεί τυχόν θόρυβος. Τέλος, μέσω μιας επαναληπτικής μεθόδου ταξινομεί τα σημεία σε “σημεία κλειδιά” και “σημεία τέλους”. Τέλος εκτελείται μια αυτόματη διαδικασία διανυσματοποίησης και διαστασιολόγησης χρησιμοποιώντας τα ταξινομημένα σημεία. Ο αλγόριθμος εφαρμόστηκε σε δεδομένα από το μνημείο Frauenkirche στη Δρέσδη.

Οι Bazazian et. al. (2015) παρουσίασαν μια τεχνική εξαγωγής ακμών χρησιμοποιώντας τα χαρακτηριστικά του πίνακα μεταβλητότητας-συμμεταβλητότητας καθώς και τις ιδιοτιμές και τα ιδιοδιανύσματα του. Ούσα μια άμεση μέθοδος, εφαρμόζεται απευθείας σε ένα δοσμένο νέφος σημείων. Για κάθε σημείο προσδιορίστηκε η “γειτονιά” του, χρησιμοποιώντας τον αλγόριθμο “KNNs”, ενώ προσαρμόστηκε σε αυτήν ένα επίπεδο, με χρήση των ελαχίστων τετραγώνων και υπολογίστηκε το κανονικό του διάνυσμα. Έχοντας όλα τα επίπεδα, τα διανύσματά τους ταξινομήθηκαν μέσω μιας συσσωρευτικής τεχνικής. Οι τελικές γραμμές προκύπτουν μέσω της συνάρτησης επιφανειακής μεταβολής από την οποία ταξινομούνται τα σημεία σε εκείνα που ανήκουν σε γραμμή και σε εκείνα που ανήκουν σε επίπεδο. Τα αποτελέσματα που παρουσιάζονται στην δημοσίευση των Bazazian et. al. (2015), δείχνουν πως κάποιες ακμές εντοπίστηκαν σωστά, ωστόσο υπάρχουν και πολλές ελλείψεις ή αστοχίες.

Οι Qi et. al. (2017) σχεδίασαν ένα καινοτόμο νευρωνικό δίκτυο το οποίο δέχεται απευθείας ένα μη-ταξινομημένο νέφος σημείων και το επιστρέφει ταξινομημένο σε ομάδες. Η μέθοδος εφαρμόζεται σε κάθε σημείο χωριστά. Αξιοσημείωτο είναι το ότι η συγκεκριμένη μέθοδος δεν μετασχηματίζει τα σημεία π.χ. σε εικόνες ή σε voxels, κατά την εκτέλεσή της. Στην πραγματικότητα προσδιορίζει μια τιμή δείκτη (label) για κάθε σημείο, δημιουργώντας ένα αραιό νέφος σημείων όπου περιγράφει το σκελετό του αντικειμένου. Τα σημεία που ανήκουν στο αραιό νέφος ονομάζονται “κρίσιμα” και χρησιμοποιούνται σαν ολικά χαρακτηριστικά σημεία. Το νευρωνικό δίκτυο διαχωρίζεται στα δύο, σε ένα που επιτελεί την ταξινόμηση και σε ένα που επιτελεί την κατάτμηση. Το κυρίως δίκτυο είναι εκείνο της ταξινόμησης, ενώ το άλλο αποτελεί μια προσθήκη. Υπάρχουν πολλές εφαρμογές στις οποίες μπορεί να χρησιμοποιηθεί η συγκεκριμένη μέθοδος όπως η κατάτμηση και ταξινόμηση νεφών που παρουσιάζουν διάφορα αντικείμενα ή χώρους.

Έμμεσες Μέθοδοι

Οι Lin Yangbin et. al. (2015) παρουσίασαν μια μέθοδο εξαγωγής 3Δ ευθυγράμμων τμημάτων χρησιμοποιώντας τομές επιπέδων. Ως δεδομένα εισόδου χρησιμοποιούνται νέφη σημείων, ενώ στη δημοσίευσή τους χρησιμοποιούν νέφη σημείων που παρήχθησαν από σαρωτή λέιζερ. Η μέθοδός τους καλείται “Line-Segment-Half-Planes” ή εν συντομία LSHP. Αρχικά το δοθέν νέφος σημείων προβάλλεται σε διάφορα επίπεδα δημιουργώντας εικόνες από διαφορετική οπτική γωνία. Στην συνέχεια εκτελείται ο αλγόριθμος LSD ο οποίος εντοπίζει ευθύγραμμα τμήματα πάνω στις παραχθείσες εικόνες. Στην συνέχεια χρησιμοποιείται ο LSHP για να αποκλείσει λανθασμένες ακμές. Ακόμα καθορίζει διάφορους γεωμετρικούς περιορισμούς σε κάθε γραμμή. Έπειτα τα εντοπισμένα σημεία προβάλλονται

στον 3Δ χώρο σε σχήμα “V”. Στην συνέχεια, εντοπίζονται τα δύο μη-παράλληλα επίπεδα στα οποία ανήκουν τα σημεία που προβλήθηκαν. Τα τελικά ευθύγραμμα τμήματα προσδιορίζονται από την τομή των επιπέδων αυτών.

Οι Xiaohu Lu et. al. (2019) δημιούργησαν μια μέθοδο εξαγωγής 3Δ ακμών από μη-οργανομένα νέφη σημείων. Αρχικά το δοθέν νέφος σημείων χωρίζεται σε μικρότερα υποσύνολα χρησιμοποιώντας τον αλγόριθμο KNN. Στα σύνολα αυτά προσαρμόζεται ένα επίπεδο και υπολογίζεται το κανονικό του διάνυσμα. Έπειτα υπολογίζεται ο πίνακας μεταβλητότητας-συμμεταβλητότητας καθώς και οι ιδιοτιμές και τα ιδιοδιανύσματά του, για κάθε υποσύνολο σημείων. Μέσω αυτών προσδιορίζονται τέσσερις χαρακτηριστικές τιμές για κάθε περιοχή. Στη συνέχεια, πραγματοποιείται μια επαναληπτική διαδικασία κατά την οποία ξεκινώντας με ένα αρχικό σημείο και τη γειτονιά-περιοχή του, ελέγχονται τα σημεία που εντάσσονται σε αυτή, κατά πόσον ικανοποιούν κάποια κριτήρια. Στην περίπτωση που τα ικανοποιούν ενσωματώνονται στην εκάστοτε δημιουργούμενη περιοχή. Έτσι η κάθε μία περιοχή μεγαλώνει έως ότου να μην υπάρχουν σημεία γύρω της, που να ικανοποιούν αυτά τα κριτήρια. Αν οι τελικές αναπτυγμένες περιοχές περιγράφουν το ίδιο επίπεδο, ενώνονται. Αφού τα επίπεδα συγχωνευτούν, όπου αυτό είναι απαραίτητο, προβάλλονται σε ένα επίπεδο δημιουργώντας μια εικόνα. Εν τέλει οι εικόνες υπόκεινται σε επεξεργασία εξαγωγής ακμών. Οι εντοπισμένες ακμές, κάθε επιπέδου, προβάλλονται στο 3Δ χώρο ενώ μια διαδικασία φιλτραρίσματος αποκλείει κάποιες λανθασμένες ακμές.

Η Dolapsaki (2020) σχεδίασε και ανέπτυξε έναν αλγόριθμο για εξαγωγή ακμών από μη ταξινομημένα νέφη σημείων χρησιμοποιώντας μια εικόνα. Η προτεινόμενη μέθοδος εφαρμόστηκε και αξιολογήθηκε πάνω σε δεδομένα πεδίου. Αρχικά ο χρήστης εντοπίζει τη ζητούμενη γραμμή πάνω στην εικόνα. Για τις ανάγκες του παραδείγματος αυτή η διαδικασία έγινε χειροκίνητα. Στη συνέχεια η ακμή προσδιορίζεται χρησιμοποιώντας την μέθοδο των ελαχίστων τετραγώνων. Στην συνέχεια, οι εικονοσυντεταγμένες υπόκεινται σε μια στροφή και μετάθεση βάσει του εξωτερικού προσανατολισμού της εικόνας ώστε να αναφερθούν στο ίδιο σύστημα συντεταγμένων με το διαθέσιμο νέφος σημείων. Έπειτα μέσω του αλγορίθμου RANSAC προσαρμόζεται στα μετασχηματισμένα σημεία μια 3Δ γραμμή, ενώ υπολογίζονται και οι μεταβλητές του επιπέδου που ορίζεται από δύο σημεία τη γραμμή και το κέντρο προβολής της εικόνας. Αμέσως μετά, υπολογίζεται η ευκλείδεια απόσταση κάθε σημείου από το επίπεδο δημιουργώντας έναν πίνακα δεδομένων, ο οποίος καθαρίζεται από το θόρυβο που περιέχει λόγω των παράλληλων, προς τη ζητούμενη ευθεία, ευθειών. Εν τέλει ο αλγόριθμος RANSAC εκτελείται ξανά με σκοπό να μειώσει την ύπαρξη τυχαίου θορύβου.

Ο Alshawabkeh (2020) ανέπτυξε μια μέθοδο εξαγωγής 3Δ γραμμών που χρησιμοποιεί συνδυαστικά δεδομένα από σαρωτές λέιζερ και εικονιστικά. Ο σκοπός ήταν ο εντοπισμός ρηγματώσεων στα μνημεία της Πέτρας στην Ιορδανία. Το νέφος σημείων μετατρέπεται σε εικόνες βάθους, οι οποίες έχουν τα ίδια χαρακτηριστικά με τις εικόνες RGB. Στη συνέχεια οι εικόνες και τα δεδομένα LiDAR μετασχηματίζονται ώστε να αναφέρονται στο ίδιο σύστημα συντεταγμένων. Έπειτα, σε κάθε εικόνα εφαρμόζεται ο αλγόριθμος Canny [Canny (1983) και (1986)] ώστε να εντοπισθούν 2Δ ακμές. Εν τέλει, οι εξαχθείσες ακμές συσχετίζονται με τα δεδομένα LiDAR και προσδιορίζονται οι 3Δ συντεταγμένες τους.

Οι Hofer et. al (2017) πρότειναν μια μέθοδο η οποία εντοπίζει και διανυσματοποιεί 3Δ γραμμές. Η μέθοδος αρχικά εντοπίζει και συνταυτίζει τις γραμμές πάνω στις δεδομένες εικόνες και εν τέλει τις προσδιορίζει στον 3Δ χώρο. Ο σκοπός αυτής της μεθόδου ήταν η παρουσία μιας εναλλακτικής, στη μέθοδο της πυκνής συνταύτισης εικόνων (MVS), υπό την έννοια ότι αποτελεί μια απλοποιημένη μέθοδο η οποία παράλληλα παρέχει με ακρίβεια την 3Δ πληροφορία της εκάστοτε σκηνής.

Ο Skentzos (2020) παρουσίασε μια αυτοματοποιημένη μέθοδο εντοπισμού ακμών σε μη ταξινομημένα νέφη σημείων. Η μέθοδος εντοπίζει τις ακμές χρησιμοποιώντας ένα βελτιωμένο αλγόριθμο Canny [Canny (1983) και (1986)]. Η βελτίωσή του έγκειται στη χρήση μιας τεχνικής αμφίπλευρου φιλτραρίσματος των δεδομένων εικόνων. Αφού εντοπισθούν οι ακμές εντάσσονται στο παραγόμενο πυκνό νέφος χρησιμοποιώντας εικόνες οι οποίες έχουν επικαλυφθεί με τους παραγόμενους, από τον Canny [Canny (1983) και (1986)], χάρτες ακμών.

Οι Wang et. al. (2015) ανέπτυξαν μια αυτοματοποιημένη μέθοδο κατά την οποία τα δοσμένα νέφη σημείων χωρίζονται σε μικρά μέρη που στην συνέχεια χρησιμοποιούνται για την δημιουργία 3Δ γεωμετρικών μοντέλων. Αυτά τα μοντέλα χρησιμοποιούνται για την παραγωγή εν τέλει των 3Δ ακμών.

Το λογισμικό PointCab () προσφέρει μια αυτόματη μέθοδο εξαγωγής 2Δ σχεδίων. Αρχικά το δοθέν νέφος σημείων μετασχηματίζεται σε τρεις εικόνες, οι οποίες παρουσιάζουν διαφορετικές όψεις του. Στη συνέχεια, χρησιμοποιώντας αυτές τις εικόνες, αρχικά εντοπίζει και εν τέλει διανυσματοποιεί τις 2Δ ακμές. Κατ' αυτό τον τρόπο επιστρέφονται στον χρήστη 2Δ σχέδια για κάθε όψη του αντικειμένου που εισήχθη.

Θεωρητική Προσέγγιση

Δομή από Κίνηση και Πυκνή Συνταύτιση

Ένα θεμελιώδες πρόβλημα της φωτογραμμετρίας και της όρασης υπολογιστών, είναι η 3Δ ανακατασκευή αντικειμένων, σκηνών κ.ά. κατά την οποία προσδιορίζεται το σχήμα και το χρώμα τους στον 3Δ χώρο. Γενικά χρησιμοποιούνται δύο ειδών τεχνικές που επιλύουν το συγκεκριμένο πρόβλημα, οι παθητικές και οι ενεργητικές. Οι παθητικές μέθοδοι χρησιμοποιούν αισθητήρες, ευαίσθητους σε ένα ορισμένο φάσμα της ηλεκτρομαγνητικής ακτινοβολίας, συνήθως στο ορατό, ώστε να ανακατασκευάσουν το εκάστοτε αντικείμενο. Στην πραγματικότητα, οι παθητικές μεθοδολογίες καταγράφουν την ανάκλαση του φωτός πάνω στην επιφάνεια των αντικειμένων, σε ένα συγκεκριμένο φάσμα της ηλεκτρομαγνητικής ακτινοβολίας. Στη συνέχεια οι παραγόμενες εικόνες χρησιμοποιούνται ως δεδομένα εισόδου στον αλγόριθμο παραγωγής του 3Δ μοντέλου. Οι ενεργητικές μέθοδοι, χρησιμοποιούν τεχνικές μέτρησης του βάθους εκμεταλλευόμενες την ταχύτητα του φωτός και τον χρόνο διαδρομής της εκπεμπόμενης ακτίνας από το εκάστοτε όργανο μέτρησης έως το σημείο, που καταγράφεται κάθε φορά και πίσω. Οι τεχνικές αυτές προσδιορίζουν άμεσα την απόσταση από κάθε σημείο (βάθος) και παράγουν απευθείας ένα νέφος σημείων αποτελούμενο από εκατομμύρια καταγραφές. Τόσο οι παθητικές όσο και οι ενεργητικές μέθοδοι είναι χρήσιμες και απαραίτητες καθώς δεν αναιρεί η μια την άλλη ενώ η επιλογή βασίζεται τόσο στο εκάστοτε αντικείμενο όσο και στον διαθέσιμο εξοπλισμό. Πιο συγκεκριμένα, σε κάποιες περιπτώσεις μπορεί να χρησιμοποιηθούν μόνο παθητικές, σε κάποιες άλλες μόνο ενεργητικές ενώ δεν λείπουν και αυτές που μπορούν να χρησιμοποιηθούν και οι δύο. Τόσο οι παθητικές όσο και οι ενεργητικές τεχνικές χαρακτηρίζονται από πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα, οι παθητικές τεχνικές καταγράφουν τα πραγματικά χρώματα του αντικείμενου, σε αντίθεση με τις ενεργητικές οι οποίες δεν καταγράφουν το χρώμα των σημείων άμεσα. Οι ενεργητικές μέθοδοι έχουν την δυνατότητα της άμεσης εφαρμογής τους σε συνθήκες χαμηλού φωτισμού, σε αντίθεση με τις παθητικές που χρειάζονται εξωτερικό φωτισμό. Στην παρούσα διπλωματική εργασία, χρησιμοποιούνται εικόνες, στο ορατό φάσμα της ηλεκτρομαγνητικής ακτινοβολίας, οι οποίες εισάγονται στον αλγόριθμο παραγωγής 3Δ μοντέλων. Ωστόσο, ο αλγόριθμος εκτελείται μέχρι και την παραγωγή του πυκνού νέφους σημείων. Στην πραγματικότητα η κατασκευή του πυκνού νέφους σημείων πραγματοποιείται μέσω των αλγορίθμων, δομής από κίνηση και πυκνής συνταύτισης εικόνων. Αρχικά, οι εικόνες προσανατολίζονται δηλαδή εξάγεται, για κάθε εικόνα, ένας πίνακας μετάθεσης και στροφής, δηλαδή η σχετική τους θέση ως προς ένα κοινό αυθαίρετο σύστημα αναφοράς. Συνήθως αυτή η σχετική θέση αναφέρεται ως προς την πρώτη εικόνα, της αλληλουχίας ενώ εξάγεται από αλγόριθμο δομής από κίνηση. Προϋπόθεση γι' αυτό το βήμα είναι η αντιστοίχιση ομολόγων σημείων σε πάνω από δύο εικόνες. Αυτή η διαδικασία πραγματοποιείται μέσω διάφορων αλγορίθμων όπως ο SIFT, A-KAZE κ.ά. σε συνδυασμό με τεχνικές συνταύτισης εικόνων. Εκτός από τον προσανατολισμό των εικόνων, ο αλγόριθμος δομής από κίνηση, δημιουργεί και ένα αραιό νέφος σημείων. Τα σημεία του αραιού νέφους προκύπτουν από την τομή τουλάχιστον τριών ακτίνων, οι οποίες ξεκινούν από το κέντρο προβολής της κάθε εικόνας, τέμνουν τα αντίστοιχα ομόλογα σημεία και εν τέλει "συναντιούνται" σε ένα σημείο το οποίο και αποτελεί το παραγόμενο σημείο στο αραιό νέφος.

Πολλοί αλγόριθμοι, δομής από κίνηση (SfM) έχουν σχεδιαστεί και υλοποιηθεί τα τελευταία χρόνια. Παρακάτω περιγράφεται η βασική διαδικασία του αλγορίθμου, η οποία αποτελείται από τρία γενικά βήματα. Το πρώτο βήμα είναι η εξαγωγή χαρακτηριστικών σημείων σε κάθε εικόνα της αλληλουχίας, το δεύτερο είναι η συνταύτιση των γειτονικών εικόνων ενώ το τρίτο είναι η λύση του συστήματος

χρησιμοποιώντας μια επαναληπτική διαδικασία βελτιστοποίησης των υπολογισμένων συντεταγμένων γνωστή και ως συνόρθωση δέσης (bundle adjustment). Οι μέθοδοι SfM χωρίζονται σε τρεις γενικές κατηγόριες (i) incremental SfM, (ii) global SfM και (iii) out-of-core SfM. Στην πρώτη κατηγορία, ο προσανατολισμός των εικόνων προκύπτει επιλύοντας αρχικά ένα ζεύγος εικόνων και στη συνέχεια προσθέτοντας μια-μια τις εικόνες της αλληλουχίας. Στην δεύτερη, οι προσανατολισμοί των εικόνων επιλύονται ταυτόχρονα. Η τρίτη κατηγορία αποτελεί μια ενδιάμεση προσέγγιση αφού αρχικά δημιουργεί διάφορες ξεχωριστές επιλύσεις (πρώτη κατηγορία) που εν τέλει τις ενώνει χρησιμοποιώντας μια γενικευμένη επίλυση (δεύτερη κατηγορία). Από τις τρεις κατηγορίες, οι προσεγγίσεις που ανήκουν στην πρώτη αποτελούν τις πιο ακριβείς, λόγω του τρόπου εφαρμογής της διαδικασίας συνόρθωσης δέσμης. Πιο συγκεκριμένα η διαδικασία βελτιστοποίησης του αποτελέσματος, στην περίπτωση του " incremental SfM", εφαρμόζεται για κάθε νέα εικόνα που εισέρχεται στο υπολογισμένο μοντέλο. Αυτή η διαδικασία ονομάζεται τοπική συνόρθωση. Ακόμα, το τελικό μοντέλο που προκύπτει επιδέχεται άλλη μια γενική συνόρθωση. Σε αντίθεση με το " incremental SfM" , οι προσεγγίσεις που ανήκουν στην κατηγορία "global SfM" επιδέχονται μόνο την γενική περίπτωση και γι' αυτό αποτελούν μεθόδους χαμηλότερης ακρίβειας. Οι πιο συχνά χρησιμοποιούμενες μέθοδοι είναι αυτές της κατηγορίας "incremental SfM" και γι' αυτό μια ανάλυση των βασικών βημάτων τους πραγματοποιείται παρακάτω.

Ένας αλγόριθμος "incremental SfM" μπορεί να χωριστεί σε δύο βήματα, (i) την αναζήτηση ομολογιών και την (ii) διαδοχική ανακατασκευή. Με την σειρά του το πρώτο βήμα χωρίζεται σε άλλα τρία, (i) την εξαγωγή χαρακτηριστικών σημείων (feature extraction), (ii) τη συνταύτιση εικόνων (feature matching) και (iii) την γεωμετρική επαλήθευση (geometric verification). Το δεύτερο στάδιο χωρίζεται σε πέντε βήματα την (i) αρχικοποίηση (initialization), την (ii) εγγραφή εικόνων (image registration), τον (iii) τριγωνισμό (triangulation), τη (iv) συνόρθωση δέσμης (bundle adjustment) και την (v) αφαίρεση θορύβου (outlier removal).

Η εξαγωγή χαρακτηριστικών σημείων είναι μια διαδικασία κατά την οποία διαχωρίζονται χαρακτηριστικές περιοχές της κάθε εικόνας από τις υπόλοιπες. Ως χαρακτηριστικές περιοχές ορίζονται οι γωνίες καθώς και περιοχές με ενιαία χαρακτηριστικά ή σχεδόν ενιαία, που διαφέρουν από εκείνα του φόντου (blobs). Κάθε τέτοια περιοχή που εντοπίζεται θεωρείται ως ένα χαρακτηριστικό σημείο (key point) της εικόνας. Κάθε χαρακτηριστικό σημείο συνοδεύεται από έναν περιγραφέα (descriptor) που δεν είναι τίποτα άλλο παρά η γύρω περιοχή του σημείου και χρησιμοποιείται ως ένα είδος "υπογραφής" του σημείου. Υπάρχει μια πληθώρα από τεχνικές που εξάγουν τα χαρακτηριστικά σημεία εικόνων τόσο γωνιών όσο και ενιαίων περιοχών όπως οι Harris Corner (Harris & Stephens (1988)), Shi-Tomasi (Shi & Tomasi (1994), Tommasini et. al., (1998)), Förstner (Förstner et. al. (2009)) και Difference of Gaussians (Lowe (1999) & (2004)).

Η συνταύτιση εικόνων αποσκοπεί στον εντοπισμό κάθε χαρακτηριστικού σημείου σε όλες τις εικόνες που εμφανίζεται. Κάθε χαρακτηριστικό σημείο πρέπει να είναι ανεξάρτητο από τυχόν αλλαγές της θέσης του προσανατολισμού και της κλίμακάς του στην κάθε εικόνα. Αυτός είναι και ο λόγος ύπαρξης των περιγραφέων τους. Οι πρώτες μέθοδοι που υλοποιήθηκαν δεν χρησιμοποιούσαν τους περιγραφείς των σημείων και γι' αυτό δεν ήταν τόσο ακριβείς όσο οι επόμενες. Γνωστές μέθοδοι συνταύτισης εικόνων είναι οι SIFT (Lowe 2004), SURF (Bay et. al. 2006), BRIEF (Calonder et. al. 2011) και ORB (Rublee et. al. 2011). Στην πραγματικότητα, τα χαρακτηριστικά σημεία αντιστοιχίζονται με εκείνα που οι περιγραφείς τους έχουν την μικρότερη ευκλείδεια απόσταση. Ουσιαστικά ενώνονται τα σημεία όπου η γύρω περιοχή τους είναι παρόμοια. Υπό αυτό το πρίσμα ορίστηκε και ένα φιλτράρισμα των συνταυτίσεων κατά το οποίο ελέγχεται η τιμή της ευκλείδειας απόστασης μεταξύ των δύο κοντινότερων χαρακτηριστικών σημείων, στο υπό επεξεργασία σημείο. Αν η απόσταση είναι παρόμοια τότε τα σημεία αποκλείονται με την σκέψη ότι ένα σημείο είναι αδύνατο να αντιστοιχίζεται με δύο (Lowe ratio). Τέτοιες περιπτώσεις υπάρχουν λόγω επαναλαμβανόμενων μοτίβων.

Η γεωμετρική επαλήθευση είναι μια από τις σημαντικότερες διαδικασίες της επιπολικής γεωμετρίας. Ο σκοπός της είναι να διαχωρίσει τις σωστές από τις λανθασμένες συνταυτίσεις. Αυτή η διαδικασία επιτελείται χρησιμοποιώντας τον αλγόριθμο RANSAC με την εφαρμογή του οποίου εκτιμάται ένα μοντέλο μετασχηματισμού για τις εικόνες. Για το σκοπό αυτό υπολογίζονται ο επιπολικός (fundamental)

ή ο δεσμευμένος επιπολικός (essential) πίνακας. Ο πρώτος υπολογίζεται στις περιπτώσεις που ο εσωτερικός προσανατολισμός των εικόνων δεν είναι γνωστός ενώ ο δεύτερος όταν είναι. Για τον υπολογισμό τους χρησιμοποιείται ο αλγόριθμος των 7 και 5 σημείων αντίστοιχα.

Η αρχικοποίηση αποσκοπεί στο να εντοπίσει ένα καλό ζεύγος εικόνων ώστε να ξεκινήσει η διαδικασία επαναληπτικής αύξησης του μοντέλου. Αυτό το βήμα είναι ένα από τα βασικότερα καθώς καθορίζει την επιτυχή έκβαση της εκτέλεσης του αλγορίθμου. Πιο συγκεκριμένα σε περίπτωση κακής επιλογής του αρχικού ζεύγους η διαδικασία μπορεί να τερματιστεί πρόωρα. Εκτός αυτού, η διαδικασία της αρχικοποίησης παίζει καθοριστικό ρόλο στην αποτελεσματικότητα και την ακρίβεια της διαδικασίας εν γένει.

Κατά την εγγραφή εικόνων υπολογίζεται ο προσανατολισμός κάθε εικόνας χρησιμοποιώντας το προηγούμενα ορισμένο ζεύγος εικόνων. Στην περίπτωση που ο εσωτερικός προσανατολισμός των εικόνων είναι άγνωστος υπολογίζεται και αυτός κατά την διάρκεια αυτής της διαδικασίας. Στην πραγματικότητα, ο σκοπός είναι να προστίθενται εικόνες στο μοντέλο που έχει δημιουργηθεί από το ζεύγος εικόνων επιλύοντας ένα PnP πρόβλημα και βελτιστοποιώντας το μέσω του αλγορίθμου RANSAC. Ως PnP πρόβλημα ορίζεται η εκτίμηση της θέσης και του προσανατολισμού της εικόνας έχοντας ως δεδομένα τον εσωτερικό της προσανατολισμό και μια σειρά από 3Δ σημεία μαζί με τα αντίστοιχα 2Δ σημεία τους πάνω στις εικόνες.

Κατά τον τριγωνισμό εντοπίζονται τα 3Δ σημεία στο χώρο χρησιμοποιώντας την θέση των καμερών καθώς και τον εσωτερικό τους προσανατολισμό. Για να προσδιοριστεί το σημείο πρέπει να χρησιμοποιηθούν 2 ή παραπάνω εικόνες. Για κάθε εικόνα προσδιορίζεται η ευθεία που περνά από το εκάστοτε 2Δ σημείο και το κέντρο προβολής της. Οι ευθείες που δημιουργούνται τέμνονται στο ζητούμενο 3Δ σημείο. Κάθε νέα εικόνα που προστίθεται αυξάνει και την πληροφορία που εισάγεται στο αραιό νέφος σημείων. Υπάρχουν πολύ αλγόριθμοι που χρησιμοποιούνται για την διαδικασία του τριγωνισμού όπως Poly [Hartley, & Sturm (1997)], Poly-Abs [Hartley, & Sturm (1997)], Mid-point [Hartley, & Sturm (1997)] και Linear-Eigen [Hartley, & Sturm (1997)]

Κατά την συνόρθωση δέσμης βελτιώνονται τόσο ο προσανατολισμός των εικόνων όσο και οι συντεταγμένες των προσδιορισμένων 3Δ σημείων, ταυτόχρονα. Κατά την διαδικασία αυτή αφαιρούνται τα λάθος προσδιορισμένα σημεία (θόρυβος) ενώ ταυτόχρονα ελαττώνεται το σφάλμα επαναπροβολής, κατά το οποίο τα 3Δ σημεία προβάλλονται πίσω στο επίπεδο της εικόνας και συγκρίνεται η νέα τους θέση με την παλιά, η διαφορά αυτή αποτελεί το ζητούμενο σφάλμα. Όσο αυτό το σφάλμα μειώνεται η ακρίβεια του τελικού προϊόντος αυξάνεται. Τέλος κατά την αφαίρεση θορύβου απομακρύνονται οι λανθασμένες παρατηρήσεις.

Το αποτέλεσμα της δομής από κίνηση, εισάγεται στην πυκνή συνταύτιση εικόνων. Η πιο γνωστή μέθοδος συνταύτισης εικόνων είναι η CMVS-PMVS (Furukawa et. al. (2010)). Αρχικά η διαδικασία CMVS παραλαμβάνει το αραιό νέφος σημείων και τις προσανατολισμένες εικόνες. Στην συνέχεια τις ταξινομεί σε ομάδες διαχειρίσιμου αριθμού εικόνων. Εκτός του αριθμού των εικόνων, που περιέχει κάθε ομάδα, ελέγχεται και το αν κάθε σημείο που είχε προκύψει κατά την διαδικασία SfM μπορεί να προσδιοριστεί τουλάχιστον από μια εκ των ομάδων. Το αραιό νέφος σημείων αντιμετωπίζεται από τον αλγόριθμο ως μια δειγματοληψία του πυκνού νέφους το οποίο πρέπει να δημιουργηθεί. Στην συνέχεια εκτελείται ο αλγόριθμος PMVS από τον οποίο προκύπτει το τελικό πυκνό νέφος σημείων.

Σημασιολογική πληροφορία

Στην επιστήμη της όρασης υπολογιστών και της επεξεργασίας εικόνας ο όρος σημασιολογική πληροφορία αναφέρεται σε μια επιπλέον τιμή, εκτός των RGB, η οποία συσχετίζει κάθε εικονοψηφίδα ή 3Δ σημείο με μια σημασία. Πιο συγκεκριμένα, τα δοθέντα δεδομένα αρχικά χωρίζονται σε ομάδες, όπου κάθε εικονοψηφίδα ή 3Δ σημείο που συμμετέχει έχει κοινά ή σχεδόν κοινά χαρακτηριστικά με τα υπόλοιπα και στη συνέχεια συσχετίζονται με μια τιμή που αντιπροσωπεύει την σημασιολογική τους πληροφορία. Το πρώτο σκέλος ονομάζεται κατάτμηση ενώ το δεύτερο ταξινόμηση ή σημασιολογική κατάτμηση. Τόσο το πρώτο όσο και το δεύτερο βήμα είναι απαιτητικές διαδικασίες για τις οποίες

προσφέρονται μια πληθώρα αλγορίθμων και εφαρμογών. Οι τεχνικές που χρησιμοποιούνται κατά τη σημασιολογική κατάτμηση μπορούν να χωριστούν στις παραδοσιακές και στις σύγχρονες. Στη συνέχεια παρουσιάζονται κυρίως κάποιες παραδοσιακές τεχνικές. Ωστόσο, γίνονται αναφορές και σε κάποιες πιο σύγχρονες.

Η κατάτμηση βάσει περιοχής χωρίζεται στην τοπική και στην ολική. Η πιο απλή εκδοχή της κατάτμησης βάσει περιοχής είναι η κατωφλίωση. Εάν χρησιμοποιηθεί μια τιμή κατωφλίωσης τότε η διαδικασία κατωφλίωσης ονομάζεται ολική, ενώ στην αντίθετη περίπτωση ονομάζεται τοπική. Κατά την ολική, τα δοσμένα δεδομένα χωρίζονται σε δύο κατηγόριες. Η πρώτη περιέχει τα σημεία που είναι μικρότερα ή ίσα με την τιμή κατωφλίωσης ενώ η δεύτερη τα υπόλοιπα. Εάν δοθούν πάνω από μια τιμές πχ. n, τότε τα δεδομένα χωρίζονται σε n+1 κατηγορίες. Μια πιο σύνθετη μέθοδος κατάτμησης είναι η προσέγγιση ανάπτυξης περιοχής (region-growth approach). Αυτές επιλέγουν κάποια σημεία αρχικοποίησης και χρησιμοποιώντας ένα κριτήριο ανάπτυξης αυξάνουν τα σημεία κάθε κατηγορίας, ενσωματώνοντας εκείνα που βρίσκονται στα όρια της κάθε περιοχής και ικανοποιούν το κριτήριο ανάπτυξης. Η μέθοδος της κατωφλίωσης είναι πιο γρήγορη σε σχέση με εκείνη της ανάπτυξης περιοχής, ωστόσο η δεύτερη δίνει συνήθως καλύτερα αποτελέσματα.

Οι μέθοδοι κατάτμησης με προσέγγιση μοντέλου, χρησιμοποιούνται πολύ συχνά όταν τα δοσμένα δεδομένα είναι 3Δ, επειδή έχουν την δυνατότητα να τα διαχωρίσουν σε κύρια σχήματα, όπως σφαίρες, επίπεδα κλπ. Σε αυτές τις μεθόδους συγκαταλέγονται ο αλγόριθμος RANSAC και ο Hough Transform. Εφαρμόζοντας αυτούς τους αλγορίθμους τα σημεία χωρίζονται σε σημεία που ικανοποιούν το μοντέλο και σε εκείνα που δεν το ικανοποιούν. Έτσι προκύπτουν διάφορες ομάδες δεδομένων που ικανοποιούν μοντέλα με διαφορετικές παραμέτρους.

Η επιβλεπόμενη και η μη-επιβλεπόμενη είναι δύο από τις πιο γνωστές μεθόδους εκμάθησης. Στην επιβλεπόμενη, ο χρήστης δίνει στον εκάστοτε αλγόριθμο μερικά παραδείγματα κατηγοριών και εν συνεχεία αυτός αντιστοιχεί όλα τα δεδομένα σε μια από τις δοσμένες κατηγορίες. Η μη-επιβλεπόμενη μέθοδος εφαρμόζεται απευθείας στα δεδομένα χωρίς να έχει δοθεί από τον χρήστη κάποιο παράδειγμα. Συνήθως δίνεται ο αριθμός των κατηγοριών όπου τα δεδομένα θα χωριστούν. Γνωστοί μη-επιβλεπόμενοι αλγόριθμοι είναι ο k-means και ο DBSCAN. Στην παρούσα διπλωματική εργασία χρησιμοποιείται ο DBSCAN.

Μια διαφορετική προσέγγιση κατάτμησης μιας εικόνας, είναι μέσω τεχνικών εξαγωγής ακμών. Σε αντίθεση με τις προηγούμενες προσεγγίσεις, που ως σκοπό έχουν τον απευθείας προσδιορισμό των ζητούμενων περιοχών, ο σκοπός των μεθόδων που χρησιμοποιούν αλγορίθμους εξαγωγής ακμών είναι ο προσδιορισμός των ορίων των περιοχών αυτών. Ως ακμές ορίζονται κυρίως ομάδες εικονοστοιχείων στα οποία η ένταση του χρώματος αλλάζει δραστικά. Ο εντοπισμός των ακμών πραγματοποιείται υπολογίζοντας είτε την πρώτη είτε την δεύτερη παράγωγο της εικόνας. Στην περίπτωση της πρώτης παραγώγου, οι θέσεις που παρουσιάζουν τοπικά μέγιστα ή ελάχιστα αποτελούν και τις θέσεις των εικονοστοιχείων που απαρτίζουν μια ακμή ενώ στην περίπτωση της δεύτερης παραγώγου οι ακμές εντοπίζονται στις θέσεις όπου οι τιμές των εικονοστοιχείων μηδενίζουν. Εκτός των παραδοσιακών τεχνικών εξαγωγής ακμών, οι οποίες χρησιμοποιούνται μέχρι και σήμερα, ο αλγόριθμος Canny [Canny (1983) και (1986)] αποτελεί την πιο συχνά χρησιμοποιούμενη μέθοδο. Στην παρούσα διπλωματική εργασία χρησιμοποιείται ο αλγόριθμος Canny [Canny (1983) και (1986)] για την εξαγωγή ακμών οι οποίες εισάγονται ως τέταρτο κανάλι στις υπό επεξεργασία εικόνες και χρησιμοποιούνται για τον εντοπισμό τους στο 3Δ χώρο.

Πρακτική Εφαρμογή

Στο πλαίσιο αυτής της διπλωματικής εργασίας χρησιμοποιήθηκαν εικόνες από δύο προηγούμενες εργασίες οι οποίες αφορούσαν στον Ναό της θεάς Δήμητρας στην Νάξο και τον Παλιό Αστυνομικό Σταθμό στο Μονόλιθο της Ρόδου. Παρακάτω δίνονται κάποιες πληροφορίες τόσο για το Ναό της Δήμητρας όσο και για τον Παλιό Αστυνομικό Σταθμό.

Ο Ναός της θεάς Δήμητρας χρονολογείται από το 525 π.Χ. χτίστηκε κατά την αρχαϊκή περίοδο ($7^{ος} - 8^{ος}$ αι. π.Χ.). Οι ανασκαφές ξεκίνησαν το 1949 και ολοκληρώθηκαν το 1995. Ο ναός βρίσκεται 11 χιλιόμετρα νοτιοανατολικά της Χώρας της Νάξου, δηλαδή περίπου στο κέντρο του νησιού. Η γεωμετρική τεκμηρίωση του πραγματοποιήθηκε για τις μεταπτυχιακές διπλωματικές εργασίες των Stefanou (2018) και Giannakoula (2018) με χρήση σύγχρονων ψηφιακών τεχνικών. Για τις ανάγκες αυτής της διπλωματικής εργασίας χρησιμοποιήθηκε μικρός αριθμός εικόνων από τον Ναό της θεάς Δήμητρας. Οι εικόνες αρχικά εμπλουτίστηκαν με την πληροφορία των ακμών τους και στην συνέχεια δόθηκαν ως δεδομένα εισόδου σε αλγορίθμους SfM-MVS για τον εντοπισμό των ακμών στο 3Δ χώρο. Ο Μονόλιθος είναι ένα χωριό της Ρόδου με περίπου 150 κατοίκους και βρίσκεται σχεδόν 72 χλμ. νοτιοδυτικά της πόλης της Ρόδου. Στην ευρύτερη περιοχή του χωριού συναντάται μεγάλος αριθμός χώρων πολιτιστικής κληρονομιάς όπως το κάστρο του Μονόλιθου ή ο αρχαιολογικός χώρος της αρχαία Κυμισάλας. Λόγω αυτών, τόσο ο Μονόλιθος όσο και τα γύρω χωριά έχουν εγείρει το ενδιαφέρον επιστημόνων από πολλούς διαφορετικούς επιστημονικούς κλάδους, όπως η αρχαιολογία, η αρχιτεκτονική και η τοπογραφία. Ο Παλιός Αστυνομικός Σταθμός στο Μονόλιθο, κατασκευάστηκε από τους Ιταλούς. Σήμερα δεν χρησιμοποιείται ως αστυνομικός σταθμός αλλά από τους αρχαιολόγους του Πανεπιστημίου του Αιγαίου, που λαμβάνουν μέρος στις αρχαιολογικές ανασκαφές στην αρχαία Κυμισάλα. Για τις ανάγκες της παρούσας διπλωματικής εργασίας χρησιμοποιείται μικρός αριθμός εικόνων του Παλιού Αστυνομικού Σταθμού που πάρθηκαν κατά την διάρκεια καλοκαιρινού μαθήματος της φωτογραμμετρίας που διεξήχθη από το εργαστήριο Φωτογραμμετρίας της Σχολής Αγρονόμων και Τοπογράφων Μηχανικών του ΕΜΠ το 2018.

Για την δημιουργία του πυκνού νέφους σημείων χρησιμοποιήθηκαν δύο γνωστά λογισμικά το Agisoft Metashape και το OpenSfM. Το πρώτο είναι ένα εμπορικό λογισμικό που παρουσιάστηκε το 2006 ως Agisoft PhotoScan. Είναι ένα από τα κορυφαία φωτογραμμετρικά λογισμικά της αγοράς και ως εκ τούτου χρησιμοποιείται σε πολλές εφαρμογές και έρευνες. Για παράδειγμα χρησιμοποιείται για την παραγωγή τρισδιάστατων μοντέλων, πυκνού νέφους σημείων, δημιουργία πανοράματος κ.ά. Επιπλέον προσφέρει την δυνατότητα διαχείρισης των λειτουργιών με χρήση προγραμματιστικών γλωσσών και πιο συγκεκριμένα της Python και της Java. Σε αυτή την διπλωματική εργασία η προσφερόμενη φωτογραμμετρική διαδικασία εκτελείται έως και τη δημιουργία του πυκνού νέφους σημείων. Εν τέλει ένα από τα θετικά του λογισμικού αυτού είναι η δυνατότητα επεξεργασίας πολυκάναλων εικόνων και όχι μόνο των κοινών RGB. Το OpenSfM είναι ένα ανοιχτό λογισμικό δομής από κίνηση και πυκνής συνταύτισης εικόνων και διανέμεται κάτω από την άδεια BSD-2-Clause. Το λογισμικό είναι διαθέσιμο στο GitHub και είναι γραμμένο σε γλώσσα Python. Μετά την εγκατάστασή του, ο χρήστης μπορεί να εκτελέσει τον αλγόριθμο SfM-MVS που παρέχεται, μέσω εντολών τερματικού. Το OpenSfM παρέχει τον πηγαίο κώδικά του και έτσι μπορεί να προσαρμοστεί στις ανάγκες του κάθε χρήστη. Ωστόσο δεν μπορεί να διαχειριστεί εικόνες με πάνω από τρία κανάλια και ως εκ τούτου στην παρούσα διπλωματική μετατράπηκε κατάλληλα για να ικανοποιεί αυτή την ανάγκη. Στα μειονεκτήματα του λογισμικού συγκαταλέγονται το ότι δεν εγγυάται το αποτέλεσμά του καθώς και το ότι ο κάθε χρήστης πρέπει να προσδιορίσει αρκετές παραμέτρους πριν την εκτέλεσή του ώστε να παραγάγει ένα ακριβές -στο πλαίσιο του δυνατού- μοντέλο. Εκτός των παραπάνω αλγορίθμων που χρησιμοποιήθηκαν για τη δημιουργία του πυκνού νέφους με σκοπό την εξαγωγή των 3Δ ακμών, χρησιμοποιήθηκε και το λογισμικό Meshroom που παρέχεται από την AliceVision κάτω από την άδεια MPL2. Το λογισμικό αυτό διαθέτει γραφικό περιβάλλον και λειτουργεί βάσει ενός δημιουργού γραφημάτων. Πιο συγκεκριμένα παρέχονται διάφοροι κόμβοι οι οποίοι αντιστοιχούν σε κάποια διαδικασία π.χ. την εξαγωγή χαρακτηριστικών σημείων, την παραγωγή πυκνού νέφους, την παραγωγή 3Δ μοντέλου κλπ. Έτσι ενώνοντας τους κόμβους μεταξύ τους ο χρήστης δημιουργεί τον εκτελέσιμο αλγόριθμο και δέχεται τα παράγωγα που επιθυμεί. Σε αυτή την διπλωματική εργασία δημιουργήθηκαν κάποια πυκνά νέφη σημείων για την διεξαγωγή διάφορων

συγκρίσεων. Τέλος για την παρουσίαση των 3Δ νεφών σημείων καθώς και για μερικές μετρήσεις χρησιμοποιήθηκε το λογισμικό CloudCompare.

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η εξαγωγή 3Δ ακμών από νέφη σημείων χρησιμοποιώντας σημασιολογική πληροφορία. Το νέφος σημείων παράγεται μέσω εικόνων δηλαδή της διαδικασίας SfM-MVS. Η σημασιολογική πληροφορία παράγεται μέσω τεχνικών εξαγωγής ακμών. Αφού προσδιοριστούν οι ακμές, κάθε εικόνα εμπλουτίζεται με την σημασιολογική της πληροφορία προσθέτοντας ένα επιπλέον κανάλι στα ήδη υπάρχοντα RGB. Οι εικόνες τεσσάρων καναλιών, που παράγονται εισάγονται σε έναν εκ των αλγορίθμων SfM-MVS. Το αποτέλεσμα είναι ένα νέφος σημείων που περιέχει τη σημασιολογική πληροφορία κάθε παραγόμενου 3Δ σημείου ως μια επιπλέον τιμή στις ήδη υπάρχουσες που αφορούν στη θέση (X, Y, Z) και το χρώμα του (R, G, B). Στην συνέχεια τα σημεία του πυκνού νέφους ταξινομούνται σε σημεία που προέρχονται από ακμές και σε όλα τα υπόλοιπα. Έπειτα τα σημεία που εντοπίστηκαν ως σημεία ακμών διαχωρίζονται σε σημεία της κάθε ακμής και διανυσματοποιούνται.

Η βασική αρχή του παραπάνω αλγορίθμου είναι ο εμπλουτισμός των εικόνων με ένα επιπλέον κανάλι και η είσοδός τους σε έναν αλγόριθμο SfM-MVS για την παραγωγή του νέφους σημείων στο οποίο έχει έτσι περάσει η σημασιολογική πληροφορία. Για τη διερεύνηση, της βασικής αρχής του παραπάνω αλγορίθμου αρχικά δημιουργήθηκε το πρόγραμμα “MyTriangulation” σε γλώσσα Python. Σε αυτό, απλοποιείται η διαδικασία SfM-MVS σε τριγωνισμό δύο εικόνων κατά την οποία παράγεται ένα μη-βέλτιστο νέφος σημείων. Το παραγόμενο νέφος σημείων ελέγχθηκε, για το αν παρουσιάζει -στο μέτρο του δυνατού- σωστά αποτελέσματα και διορθώθηκε, συγκρίνοντάς το με νέφη που προέκυψαν χρησιμοποιώντας τις ίδιες δύο εικόνες και τα λογισμικά Metashape και Meshroom. Ακόμα ελέγχθηκε για την παραγωγή των χρωμάτων του χρησιμοποιώντας μια εικόνα με ευδιάκριτες αλλαγές στα χρώματά της.

Η πρώτη εφαρμογή διερεύνησης με χρήση σημασιολογικής πληροφορίας, πραγματοποιήθηκε με εικόνες από τον αρχαιολογικό χώρο της αρχαίας Κυμισάλας. Αρχικά δημιουργήθηκαν νέες εικόνες, με βάση τις παλιές, πάνω στις οποίες είχαν χρωματιστεί με διάφορα χρώματα, κάποιες από τις πέτρες που απεικονίζονταν. Κατά την δημιουργία του νέφους χρησιμοποιήθηκαν οι αυθεντικές εικόνες για τον υπολογισμό των σημείων ενώ για τα χρώματα οι νέες. Έτσι τα σημεία κάθε πέτρας, στο νέφος σημείων, μπορούσαν να εξαχθούν βάσει του χώματος που είχε η πέτρα στην νέα εικόνα. Ωστόσο αυτή η προσέγγιση εκτός του ανακριβούς νέφους σημείων που προκύπτει, χάνει και την πληροφορία του χρώματος αφού για το χρωματισμό των σημείων χρησιμοποιούνται οι νέες εικόνες. Έτσι διερευνήθηκε ο εμπλουτισμός των εικόνων με ένα νέο κανάλι που θα περιέχει την σημασιολογική πληροφορία. Για το σκοπό αυτό δημιουργήθηκε το πρόγραμμα “SemanticPass”. Αρχικά, διαβάζει τόσο τις πρώτες εικόνες όσο και τις εικόνες χρωματισμού. Στην συνέχεια διαχωρίζει τα κανάλια κάθε μιας και εντοπίζει την σημασιολογική πληροφορία. Έπειτα δημιουργεί το νέο κανάλι “L” και το συγχωνεύει με τα RGB της εκάστοτε αυθεντικής εικόνας. Τέλος αποθηκεύει τις RGBL εικόνες. Αυτές οι εικόνες εισάγονται στο πρόγραμμα “MyTriangulation” και παράγεται το νέφος σημείων. Πλέον το νέφος δεν περιέχει μόνο τις τιμές XYZ και RGB για κάθε σημείο, αλλά και την τιμή L. Η νέα προσέγγιση εφαρμόστηκε σε δύο εικόνες από τον αρχαιολογικό χώρο του Ναού της θεάς Δήμητρας. Αρχικά δημιουργήθηκαν δύο νέες εικόνες, με βάσει τις αυθεντικές, στις οποίες δύο από τις ακμές του Ναού είχαν ζωγραφιστεί κόκκινες. Οι αυθεντικές εικόνες καθώς και οι “χρωματισμένες” εισήχθησαν στο πρόγραμμα “SemanticPass”. Αρχικά παρήχθησαν δύο μάσκες, μια για κάθε εικόνα που είχαν την τιμή 255 στα εικονοψηφία που αφορούσαν τις ακμές και την τιμή 0 σε όλα τα υπόλοιπα. Στην συνέχεια συνδυάστηκαν με τις αυθεντικές εικόνες και δημιουργήθηκαν οι τετρακάναλες. Οι εμπλουτισμένες εικόνες δόθηκαν στον αλγόριθμο “MyTriangulation” και προέκυψε το νέφος σημείων. Τα σημεία που ανήκαν στις γραμμές εξήχθησαν και παρουσιάστηκαν. Αυτή η εφαρμογή διόρθωσε το πρόβλημα της απώλειας των πραγματικών χρωμάτων. Ωστόσο το παραγόμενο νέφος συνέχισε να είναι προϊόν δύο εικόνων και επομένως αμφιβόλου ποιότητας. Για τον λόγο αυτό διερευνήθηκε ο συνδυασμός της μεθόδου με αξιόπιστους αλγορίθμους εξαγωγής πυκνών νεφών σημείων και πιο συγκεκριμένα των λογισμικών OpenSfM και Metashape.

Το λογισμικό OpenSfM δεν έχει την δυνατότητα επεξεργασίας εικόνων πολλών καναλιών. Γι’ αυτό το λόγο ο πηγαίος κώδικας τροποποιήθηκε ώστε να διαβάζει τις εικόνες αυτές και να περνά στο παραγόμενο νέφος την σημασιολογική πληροφορία. Στην συνέχεια έγιναν κάποιες εφαρμογές χρησιμοποιώντας το

τροποποιημένο OpenSfM. Σε κάθε εφαρμογή οι παράμετροι του αλγορίθμου άλλαζαν. Για παράδειγμα έγιναν εφαρμογές χρησιμοποιώντας τον αλγόριθμο SIFT αλλά και τον αλγόριθμο A-KAZE. Αξίζει να σημειωθεί ότι κατά την τροποποίηση χρησιμοποιήθηκε το σύστημα διαχείρισης-εκδόσεων GIT ώστε να υπάρξει μια λειτουργική διαχείριση των αλλαγών χωρίς απώλειες. Τα αποτελέσματα ήταν σαφώς καλύτερα σε σχέση με τις πρώτες εφαρμογές. Στη συνέχεια πραγματοποιήθηκαν εφαρμογές χρησιμοποιώντας τον αλγόριθμο Canny [Canny (1983) και (1986)] για την εξαγωγή των ακμών. Πιο συγκεκριμένα, για ένα χρήστη είναι σχεδόν αδύνατο να χρωματίσει όλες τις γραμμές, για παράδειγμα του ναού της Δήμητρας, σε όλες τις εικόνες όπου αυτές απεικονίζονται. Έτσι ο αυτοματοποιημένος εντοπισμός των ακμών σε εικόνες μέσω του αλγορίθμου Canny [Canny (1983) και (1986)] δίνει την δυνατότητα εντοπισμού περισσότερων ακμών. Η διαδικασία παρέμεινε ίδια, απλώς αντί για να δίνεται μια εικόνα χρωματισμού από την οποία να παράγεται η μάσκα της αυθεντικής εικόνας, δόθηκε απευθείας ως μάσκα ο χάρτης ακμών που παρήχθη. Μετά την παραγωγή του νέφους σημείων, εντοπίστηκαν τα σημεία που προέρχονταν από ακμές, διαχωρίστηκαν σε σημεία της κάθε ακμής και εν τέλει διανυσματοποιήθηκαν.

Το λογισμικό Metashape έχει την δυνατότητα επεξεργασίας εικόνων με παραπάνω από τρία κανάλια. Έτσι οι εικόνες που δημιουργήθηκαν στις προηγούμενες εφαρμογές, εισήχθησαν στον αλγόριθμο του λογισμικού και υπολογίστηκε το πυκνό νέφος σημείων. Το νέφος που προέκυψε υπέστη την ίδια επεξεργασία με εκείνο από το OpenSfM. Στην συνέχεια δημιουργήθηκε ένα πρόγραμμα σε Python “MetaSfM” το οποίο υλοποιεί τη διαδικασία του λογισμικού Metashape μέσω της γλώσσα Python και του πακέτου που προσφέρει η Agisoft. Έτσι πραγματοποιήθηκε μια εφαρμογή χρησιμοποιώντας το πρόγραμμα “MetaSfM”. Τόσο η εφαρμογή του Metashape μέσω του γραφικού του περιβάλλοντος όσο και με το πρόγραμμα Python έδωσαν πολύ ικανοποιητικά αποτελέσματα.

Οι αλγόριθμοι που αναπτύχθηκαν συνδυάστηκαν κάτω από το πρόγραμμα “3DPlan” που έχει ως σκοπό την εύκολη και αυτόματη διαχείρισή τους από τον χρήστη. Ακόμα προσθέτει μια βελτίωση για την παραγωγή της σημαντικής σημασιολογικής πληροφορίας. Πιο συγκεκριμένα εισάγει μια επεξεργασία πραγματικού χρόνου κατά την οποία ενεργοποιούνται δύο παράθυρα, ένα για την εικόνα και ένα για το παραγόμενο χάρτη ακμών (Arapellis (2020)). Στη συνέχεια ο χρήστης ορίζει τις παραμέτρους του αλγορίθμου Canny [Canny (1983) και (1986)] βλέποντας σε πραγματικό χρόνο τις αλλαγές στο χάρτη ακμών. Εν τέλει επιλέγει τις καλύτερες γι’ αυτόν τιμές, για κάθε εικόνα ενώ παράλληλα παράγονται οι εμπλουτισμένες εικόνες. Οι εικόνες που παράγονται δίνονται σε όποιον αλγόριθμο έχει επιλέξει ο χρήστης ώστε να εφαρμοστεί η διαδικασία της δομής από κίνηση και της πυκνής συνταύτισης. Το παραγόμενο νέφος σημείων εισάγεται στη διαδικασία της ταξινόμησης όπου τα σημεία χωρίζονται σε σημεία ακμών και στα υπόλοιπα. Τα σημεία που εξάγονται εισάγονται στη διαδικασία ομαδοποίησης κατά την οποία διαχωρίζονται σε σημεία της κάθε ακμής μέσω των αλγορίθμων DBSCAN και RANSAC. Κάθε ομάδα που εντοπίζεται διανυσματοποιείται και προστίθεται στο τελικό .dxf αρχείο. Το παραγόμενο προϊόν περιέχει όλες τις 3Δ διανυσματοποιημένες γραμμές.

Table of Images-Figures

Images

State of the Art:

Image 1: A pair of planes which were detected 21
Image 2: Final extracted lines 22
Image 3: Dresden Frauenkirche (left), horizontal profile (right) 22
Image 4: Eigenvalue analysis to extract sharp features 23
Image 5: a) Part Segmentation b) Critical points and Upper bound shape c) Semantic Segmentation 24
Image 6: Point cloud and the produced 3D lines 24
Image 7: Results on Castle and St Sulpice datasets 25
Image 8: User's input 25
Image 9: Detected line presented by red color and the ground truth line presented by green color 26
Image 10: a) Detected Features on images b) Detected Features on LiDAR meshed model 26
Image 11: Line3D++ results 27
Image 12: A triplet of images which are is used in SfM-MVS workflow 27
Image 13: The produced dense point cloud 28
Image 14: a) Point Cloud b) Segmented Point Cloud c) Boundaries and Edges d) Edited rough model e) Created Semantic model 28
Image 15: PointCab's vectorizer tools 29
Image 16: Detected lines with 0.5m and 163 threshold value 29
Image 17: Output in .dxf format with 0.5 m and 163 threshold value 30
Image 18: Detected lines with maximum threshold value 30
Image 19: Output in .dxf format with maximum threshold value 30

Practical Implementation:

Image 1: Four sample images of the Demeter temple dataset 60
Image 2: Old Police Station front view (© Google Earth) 62
Image 3: Proposed Approach 64
Image 4: Image Pair for "MyTriangulation" experiment 65
Image 5: Camera Matrix 65
Image 6: MyTriangulation point cloud using SIFT 66
Image 7: Image with manually added semantic information 66
Image 8: Point cloud with colored elements 67
Image 9: 3D points detection using color value 67
Image 10: MyTriangulation point cloud color check 67
Image 11: Point Clouds using professional software 68
Image 12: Corrected "MyTriangulation" point cloud using A-KAZE 68
Image 13: Semantic point cloud's registered values 70
Image 14: Enriched point cloud analysis 70
Image 15: OpenSfM default parameters 71
Image 16: OpenSfM's point cloud using the default parameters 71
Image 17: Parameters which was used with A-KAZE algorithm and OpenSfM 71
Image 18: OpenSfM's point cloud using AKAZE 72
Image 19: Parameters which was used with SIFT 72
Image 20: OpenSfM's point cloud using SIFT 72
Image 21: RGB images and manually annotated ones 74
Image 22: Enriched point cloud visualizations 74

Image 23: The extracted edge points using OpenSfM 75
Image 24: Inliers using RANSAC 76
Image 25: 3D Line using OpenSfM 76
Image 26: Edge labels using Canny 77
Image 27: Detected edges using the OpenSfM and the Canny algorithm 78
Image 28: Edge points detection using Agisoft-Metashape 79
Image 29: 3D Line using Agisoft Metashape 79
Image 30: Further classification of the detected edge points 81
Image 31: Produced point clouds using Canny and Metashape GUI 82
Image 32: Detected edge points using Canny and Metashape GUI 82
Image 33: Detected lines' classes using Metashape GUI 83
Image 34: Noise points using Metashape GUI 83
Image 35: 3D vectors using Canny and Metashape GUI 84
Image 36: Produced point clouds using Canny and Agisoft Metashape python module 84
Image 37: Detected edge points using Canny and Metashape python module 85
Image 38: Detected lines' classes using Metashape python module 85
Image 39: Noise points using Metashape python module 85
Image 40: 3D vectors using Canny and Metashape python module 86
Image 41: Canny's windows 88
Image 42: Messages and errors using the 3DPlan script 89
Image 43: Align and build dense cloud parameters 92
Image 44: Proposed approach Temple of Demeter point clouds 92
Image 45: Detected edges Temple of Demeter 93
Image 46: Line classes Temple of Demeter 94
Image 47: Noise points Temple of Demeter 95
Image 48: Vectorized lines with sparse point cloud Temple of Demeter 97
Image 49: Proposed approach Police station point clouds 97
Image 50: Detected edges Police Station 98
Image 51: Line classes police station 99
Image 52: Noise points police station 100
Image 53: Vectorized lines with sparse point cloud police station 100

Figures

State of the Art:

Figure 1: PointNet Architecture 23

Theoretical Approach:

Structure from Motion:

Figure 1: Incremental Structure-from-Motion Pipeline [Schönberger (2016)] 33
Figure 2: Gaussian-Scale-Space and Difference of Gaussian [Lowe 2004] 37
Figure 3: Maxima Minima Detection [Lowe 2004] 37
Figure 4: Image Gradients to Key Point descriptor [Lowe 2004] 38
Figure 5: Standard epipolar geometry set-up 39
Figure 6: Clustering View Algorithm step 43

Semantic Information:

Figure 1: Step edge 50
Figure 2: Noise step edge 50

FIGURE 3: A) ROOF EDGE OR LINE PROFILE B) T-JUNCTION 51
FIGURE 4: CANNY PIPELINE 53
FIGURE 5: Θ DIRECTION CLASSES 53

Practical Implementation:

FIGURE 1: NAXOS ISLAND AND THE TEMPLE OF DEMETER (© GOOGLE EARTH) 59
FIGURE 2: RHODE ISLAND AND MONOLITHOS VILLAGE (© GOOGLE EARTH) 61
FIGURE 3: MONOLITHOS VILLAGE AND OLD POLICE STATION (© GOOGLE EARTH) 61
FIGURE 4: 3DPLAN FLOWCHART 90
FIGURE 5: DIRECTORY'S STRUCTURE 91

Conclusions:

FIGURE 1: CANNY EDGE DETECTION VISUALIZATION 102
FIGURE 2: DETECTED EDGES WITH SEVERAL LABEL THRESHOLD VALUES 104
FIGURE 3: 3D PLAN WITH SEVERAL LABEL THRESHOLD VALUES 106
FIGURE 4: STONE WALL EDGE DETECTION 106
FIGURE 5: THE DENSE CLOUD AND THE 3D DETECTED EDGES 107

Tables

TABLE 1: COMPARISON BETWEEN THE IMPLEMENTED EDGE EXTRACTION TECHNIQUES 65
TABLE 2: PROPOSED APPROACH TEMPLE OF DEMETER PARAMETERS 93
TABLE 3: PROPOSED APPROACH POLICE STATION PARAMETERS 98

Abbreviations

AI: Artificial Intelligence
API: Application Programming Interface
BA: Bundle Adjustment
BRIEF: Binary Robust Independent Elementary Features
CAD: Computer-Aided Design
DBSCAN: Density-Based Spatial Clustering of Applications with Noise
DL: Deep Learning
DoG: Difference of Gaussian
E: East
GDBSCAN: Generalized Density-Based Spatial Clustering of Applications with Noise
KNN: k-Nearest Neighbors
LoG: Laplacian of Gaussian
ML: Machine Learning
MR scan: Magnetic Resonance scan
MVS: Multi View Stereo
N: North
NE: North-East
NLP: Natural Language Process
NTUA: National Technical University of Athens
NW: North-West
ORB: Oriented FAST, Rotated BRIEF
PnP: Perspective-n-Point
S: South
SE: South-East
SfM: Structure from Motion
SIFT: Scale Invariant Feature Transform
SRSE: School of Rural and Surveying Engineering
SURF: Speed Up Robust Feature
SVD: Singular Value Decomposition
SW: South-West
UAVs: Unmanned Aerial Vehicles
W: West

0. Introduction - Scope of Study

In our era, the technological progress, in the field of photogrammetry and computer vision in combination with the rapid improvement of computer hardware approaches, provide solutions for automating even more perplexing and demanding tasks. Photogrammetry is the science and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring, and interpreting photographic images and patterns of electromagnetic radiant imagery and other phenomena [ASPRS online Archived]. Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos [Wikipédia, (2020) Computer Vision]. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do [Ballard & Brown (1982), Huang (1996), Sonka, Hlavac & Boyle (2014)].

There is a variety of products that photogrammetric procedures can provide like meshes, orthophotos, orthophotomaps and 3D - 2D products in general. Depending on the application, a different product type is selected, for instance meshes are used to describe surfaces while orthophotos and orthophotomaps are raster products which combine the image's visual information with the ability to perform measurements on them. Furthermore, 3D or 2D architectural drawings are vector data which are widely recognizable and convenient to be used in many tasks, from a diversity of scientific fields.

An architectural drawing is a technical drawing of a building (or building project) that falls within the definition of architecture [Wikipédia, (2020) Architectural drawing]. The production of architectural drawings is a demanding process due to the architectural conventions that must be taken into account. Architectural 2D drawings, such as floor plans, site plans, elevations, and cross sections [Wikipédia, (2020) Architectural drawing] of an infrastructure, are useful products for engineers and a variety of scientists but also a powerful method to geometrically document a place in a widely recognizable manner. In fact, 2D and 3D drawings are still the most common technical documentation products. Additionally, 2D and 3D drawings are an efficient and realistic visualization method, which provides an effective way to communicate projects and concepts among the stakeholders. Traditionally, drawings were made in ink on paper or a similar material, and any copies required had to be laboriously made by hand [Wikipédia, (2020) Architectural drawing]. Due to the technological evolution, during the 20th and 21st centuries, the architectural drawings were passed from analogical to digital implementation. To this end, the traditional design techniques have been replaced by C.A.D technology which involves the use of computers (or workstations) to aid in the creation, modification, analysis, or optimization of a design [Sarcar, Rao, & Narayan (2008)]. C.A.D technology is used from a plethora of science branches and industries. Due to C.A.D usage, the productivity in several areas such as architecture, and mechanical engineering design, was significantly increased. In our era, the production of 2D or 3D drawings, is a manual, time consuming and tedious, process while it is still -and will continue to be- necessary for many tasks like documentation and preservation or conservation of cultural heritage, especially for tasks in the field.

It may be concluded from the foregoing that, the need for research, in the topic of 2D – 3D drawings and in general in architectural photogrammetry, is unequivocally necessary and useful over time. In fact, architectural drawings survived over the years due to their necessity to a variety of applications in addition with their convenience in use.

Cultural heritage is the legacy of physical artifacts and intangible attributes of a group or society that is inherited from past generations [Wikipédia, (2020) Cultural heritage]. Not all legacies of past generations are "heritage", rather heritage is a product of selection by society [Logan (2007)]. Additionally, current generations are obliged to protect cultural heritage and bequeath it to future generations, this action is called conservation or preservation. Cultural heritage contains tangible culture such as buildings, monuments, and books; intangible culture such as language and tradition; and natural heritage for instance culturally significant landscapes, and biodiversity [Sullivan (2016)]. Cultural heritage monuments, in most cases, are characterized by complex surfaces with a variety of materials and with

restrictions that occur due to the monument's accessibility and environment difficulties. In this study, tangible cultural heritage, especially cultural heritage monuments i.e., the Ancient Temple of Demeter on Naxos, and the Old Police Station in Monolithos Rhodes are used as examples to perform and evaluate the proposed methodology, analyzed in the present work.

One of the main tasks of the signal and image processing scientific area is dealing with edge detection in 2D images. The edge detection task has aroused the interest of numerous researchers from the very early years of the image processing field. Image edges are defined as a set of points at which the illumination is changed drastically i.e., discontinuities are present. Those discontinuities provide a useful and essential knowledge for the image scene such as depth changes and orientation, reflectance and shape of the objects depicted.

In fact, edge detection algorithms are used for a plethora of applications. Firstly, edge detection algorithms are used in robotics vision, due to the vital need of understanding the environment around them. Hence, the objects' orientation and shape can provide the appropriate information for the task of understanding the environment around autonomous vehicles (cars, ships, UAVs, planes etc.). For instance, an autonomous car must extract the road's limits or road's lanes from the acquired images, in real-time processing. For this task, an edge detection methodology in combination with A.I techniques, are performed. In addition, edge detection methods are used for medical image detection and classification tasks, such as the detection of a brain tumor from patient's MR scan [Abdel-Gawad, Said & Radwan (2020)]. Furthermore, for fingerprint recognition tasks, edge detection algorithms, like Canny [Canny (1983) and (1986)], are performed [Velapure & Talware (2020)]. In this study, the Canny edge detection algorithm is applied on each acquired image, to produce the semantic information i.e., edges, for each pixel.

The principal purpose of this diploma thesis is to detect and vectorize edges in unorganized point clouds using semantic information, for automated 2D or 3D drawing construction. Unorganized point clouds are sets of 3D points without a specific order. Point clouds are produced from digital images, through the SfM-MVS workflow, or from laser scanners with scanning implementation. Both methodologies are useful and, depending on the application, one or both can be implemented. For instance, image-based approach is not efficient with e.g., textureless or transparent objects or in low-light level conditions, while scanning approaches are not efficient with e.g., absorbing marble or black surfaces. Also, by definition, image-based approach contains the real color of each point, in contrast with laser scanner approaches. To confront this drawback most of the laser scanners have a camera, albeit not of high resolution, to concatenate the real colors with the acquired points. Semantics is the study of meaning. The term can be used to refer to subfields of several distinct disciplines including linguistics, philosophy, and computer science [Wikipédia, (2020) Semantics]. Hence, by linking each pixel with an edge semantic information, an unorganized point cloud can be classified into those points, that describe or belong a line, and all the rest. Furthermore, the classified point cloud constitutes a vital step, for a future automated workflow implementation for 2D - 3D drawing construction.

This diploma thesis investigates the combination of various SfM-MVS algorithms using suitably enriched images i.e., images with semantic information, for edge detection in the produced point cloud. This survey introduces a methodology to enrich the images with edge semantic information. At the same time, it aims to provide a clear and informative view of the methodology's vital role in the point cloud edge detection task. In addition, it will try to modify a known SfM-MVS algorithm, to use the provided enriched images in the SfM-MVS workflow, combining the pixel's semantic information with the produced 3D points. In addition, commercial SfM-MVS software are used without modifying the source code due to their terms of use and privacy policy. Then a point cloud classification is applied and the points that describe a line are detected and stored. Finally, a clustering and vectorizing approach is implemented and a ".dxf" archive which contains the 3D drawing is exported.

The structure of this thesis is presented below. Firstly, similar state-of-the-art approaches are investigated. This aims at a complete understanding of the existing efforts and methods. Then a

theoretical approach, describing the SfM-MVS procedure, as well as the terms “semantic information” and “edge semantic information”, is included. The objective of this analysis is to totally understand the known base that is applied in this study. Furthermore, the developed approaches are presented and implemented using the Ancient Temple of Demeter, Naxos dataset. Afterwards, the proposed approach is presented, while it is applied on the Ancient Temple of Demeter dataset as well as on the Old Police Station one. Finally, each methodology’s step performance is evaluated, providing useful information for the functionality of the workflow and for future additions and improvements that could be included.

References:

Abdel-Gawad, A.H., Said, L.A. and Radwan, A.G., 2020. Optimized Edge Detection Technique for Brain Tumor Detection in MR Images. *IEEE Access*, *8*, pp.136243-136259.

Sullivan, A.M., 2016. Cultural Heritage & New Media: A Future for the Past, 15 J. Marshall Rev. Intell. Prop. L. 604 (2016). *The John Marshall Review of Intellectual Property Law*, *15*(3), p.11.

ASPRS online Archived May 20, 2015, at the Wayback Machine

Ballard, D.H. and Brown, C.M., 1982. Computer vision. englewood cliffs. *J: Prentice Hall*.

Huang, T., 1996. Computer vision: Evolution and promise.

Logan, W.S., 2007. Closing Pandora's box: human rights conundrums in cultural heritage protection. In *Cultural heritage and human rights* (pp. 33-52). Springer, New York, NY.

Sonka, M., Hlavac, V. and Boyle, R., 2014. *Image processing, analysis, and machine vision*. Nelson Education.

Sarcar, M. M. M., Rao, K. M., & Narayan, K. L. (2008). *Computer aided design and manufacturing*. PHI Learning Pvt. Ltd..

Velapure, A. and Talware, R., 2020. Performance Analysis of Fingerprint Recognition Using Machine Learning Algorithms. In *Proceedings of the Third International Conference on Computational Intelligence and Informatics* (pp. 227-236). Springer, Singapore.

Wikipédia, (2020) Architectural drawing [online]. Available at: https://en.wikipedia.org/wiki/Architectural_drawing (Accessed: 30 November 2020)

Wikipédia, (2020) Computer Vision [online]. Available at: https://en.wikipedia.org/wiki/Computer_vision (Accessed: 30 November 2020)

Wikipédia, (2020) Cultural heritage [online]. Available at: https://en.wikipedia.org/wiki/Cultural_heritage (Accessed: 30 November 2020)

Wikipédia, (2020) Semantics [online]. Available at: https://en.wikipedia.org/wiki/Semantics (Accessed: 30 November 2020)

1. State of the art

There is a plethora of applications in which edge detection procedure is involved e.g., robotics, fingerprint recognition, lane detection etc. Those applications use edge detection techniques on images and then exploit the produced information to understand the scene's characteristics such as the object's shape and contour. The general idea, which this diploma thesis is involved with, is the 3D edge detection i.e., in point clouds, for automated 2D – 3D drawing construction. In the literature, the 3D edge detection techniques are abstracted into two main categories the (I) Direct and (ii) Indirect ones. Both are useful approaches while both or each of them can be applied according to the application's given data. Further, the scientific community has not proposed a fully automatic methodology for 3D edge vectorization yet, in comparison with the undoubtable need from a diversity of scientific fields. This diploma thesis investigates several procedures that can be used as part of such a methodology.

1.1. Direct Methods

Direct methods are used when images are not available e.g., in the case of point clouds produced by laser scanners. Firstly, the object of interest is isolated from the unnecessary parts or cleaned from the random noise present. Secondly, the object's planes are decomposed into n-separated planes e.g., using RANSAC algorithm. Thirdly, lines are detected using the detected planes e.g., as planes' intersections.

Mitropoulou (2017) and Mitropoulou & Georgopoulos (2019) presented an approach which detects the points, that describe an edge, in an unorganized point cloud, via plane intersection. The performance of the proposed method was evaluated and compared using real-world data of tangible cultural heritage. To this end, a part of the nave of the Hephaestus temple (Ancient market of Athens) point cloud, was used. This point cloud was produced using a 3D acquisition device and specifically by laser scanning implementation using a Leica ScanStation 2. The point cloud was characterized by ±5-6 cm uncertainty while the scanning was performed with 4mm point density. Firstly, three points are chosen randomly, from the point cloud. Every point in the point cloud that satisfies a threshold value t, is characterized as a point of the plane. These two steps are applied iteratively from all the point triplets of the given point cloud. The plane with most inliers is chosen in order to compute the optimal plane parameters. Two of the detected planes are depicted in Image 1 [Mitropoulou (2017) and Mitropoulou & Georgopoulos (2019)]

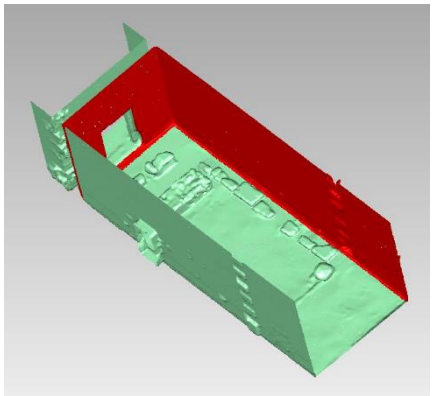


Image 1: A pair of planes which were detected

The edge is detected as an intersection of two non-parallel planes. As a rule, a line is defined from a point and a parallel to the line vector. To find the line's points in the point cloud space, Lagrange multiplier with two constraints is performed. To specify the parallel vector, the plane's exterior product is calculated. Finally, the edge's points are extracted in such a manner as to enable their immediate usage. The extraction procedure is based on the vector projection methodology. The detected 3D edges are depicted in Image 2 [Mitropoulou (2017) and Mitropoulou & Georgopoulos (2019)]

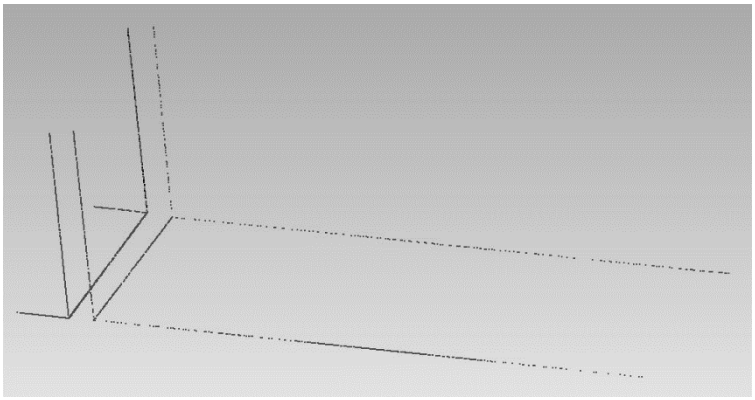


Image 2: Final extracted lines

Bienert (2008) proposed an approach of edge vectorization and dimension of smoothed point cloud profiles produced using laser scanners. Firstly, the profiles are extracted using sections, i.e., cutting the point cloud with planes e.g., a parallel to x axis or parallel to y axis with a specific thickness. The produced profiles include outlier points and so a smoothing procedure is performed using line or arc fitting elements. Line fitting element smoothing process, use an iterative procedure, with characteristics i.e., size of line element and the tolerance value, in respect to the accuracy, and the resolution of the used point cloud. According to this procedure, a line is located on one of the profile's points and then is rotated iteratively, using a pre-defined angle difference. In every iteration, each point's perpendicular to the line distance is calculated. If this distance satisfies the pre-defined tolerance value, the point is characterized as point of the line. The iteration with the most inliers, is chosen for each under process point. Afterwards, exploiting each point's slope value, due to the implementation of the previous steps on each point separately, a key point is extracted. During the procedure, several key points are produced thus some quality indices can be calculated and valuable results can be derived. These key points are obtained by detecting significantly value changes among the compared slope values during the procedure. The step before the vectorization and dimension, is the end point detection. Each under process point firstly characterized as end point while is marked with a flag. Then this assumption is checked by calculating the angle between the line, through the under process point and each of the unprocessed points, and the line produced with the most far already processed point. If this angle satisfies a threshold value i.e., smaller than 90^{o}, then the point is characterized as end point. Finally, an automated vectorization and dimension methodology using the previously determined key points and end points, is introduced. The proposed methodology was evaluated to a tangible heritage monument and more specific to Dresden Frauenkirche. The results are displayed in Image 3 [Bienert (2008)].

Image 3: Dresden Frauenkirche (left), horizontal profile (right)

Bazazian et. al. (2015) introduced an edge extraction technique using covariance matrix and their eigenvalues and eigenvectors. The proposed method is applied directly to the given point cloud. Firstly, k-near neighbors (kNNs) are estimated for each point, in the given point cloud and a plane is fitted using least squares. The normal vector is produced using neighborhood's covariance matrix. In fact, the

covariance's eigenvector that corresponds to the smaller eigenvalue constitutes the aimed normal vector. When all the normal vectors of the points are produced, new kNNs are extracted and new normal vectors are calculated. Afterwards, the normal vectors are classified using the agglomerative technique [Bazazian et. al. (2015)]. Further, Bazazian et. al. (2015), introduce an alternative, to the previously described approach, statistical method which objects to efficiently extract sharp edges. In the statistical method, since the covariance matrix has already been calculated the eigenvalues are also found. Finally, using the surface variation formula, it is determined if a point lies on an edge or on a plane. It is worth noting that, the described method's results, missing some of the major object's lines. An advantage of the proposed method is its extremely fast process. The results of the eigenvalue analysis method are depicted in Image 4 [Bazazian et. al. (2015)].

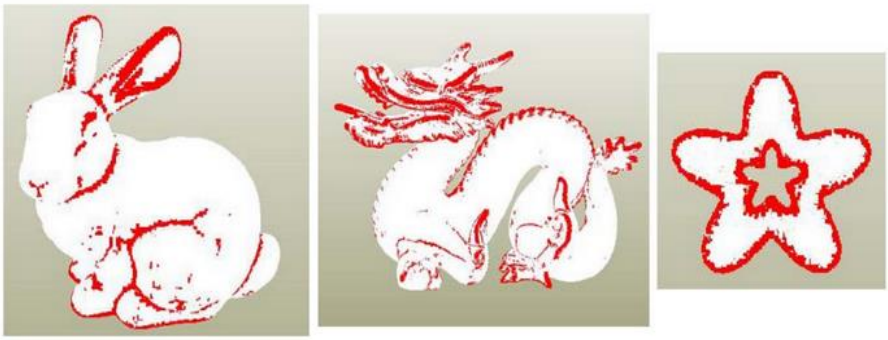


Image 4: Eigenvalue analysis to extract sharp features

Qi et. al. (2017) designed an innovative neural network which is fed directly with an unorganized point cloud and returns either labels for each point or labels for each area of points. The proposed processing is implemented on each point separately. This technique is useful for various applications like point cloud classification, part segmentation and semantic segmentation, while tackling those problems using directly the point cloud raw data i.e., without to transform it to other formats like voxels or to construct new images from it. In fact, the proposed neural network identifies the label for each point by detecting a set of sparse points which describes the skeleton of the under-process object. The detected sparse points are called critical points and behave as global features. The presented architecture is separated into two main networks the classification and segmentation. The segmentation network constitutes an add-on the main classification one. PointNet's architecture is illustrated in Figure 1 [Qi et. al. (2017)].

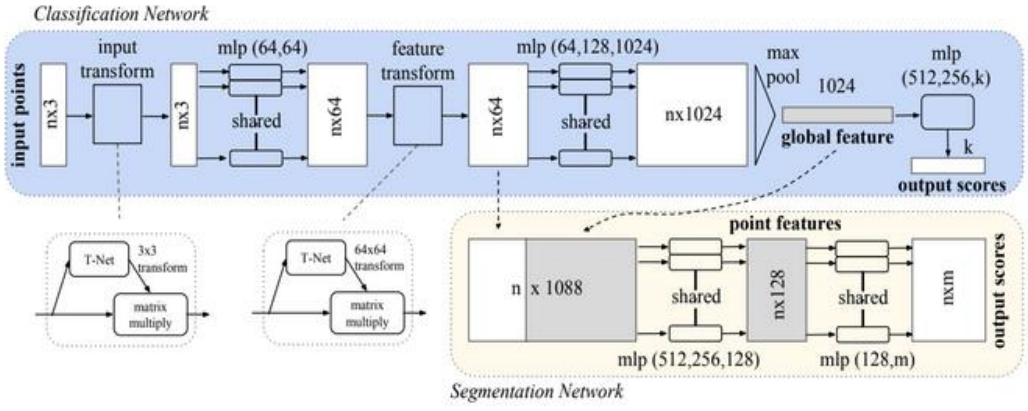


Figure 1: PointNet Architecture

The above architecture, learnt to identify the interesting and informative points i.e., skeleton or critical points, in a given point cloud using a set of optimization functional criteria. In fact, those points summarize the point cloud into a smaller set of points. PointNet's qualitative performance on various point clouds either for the part segmentation or for semantic segmentation tasks, and a visualization of critical points; and upper bound shape, are depicted in Image 5a, 5c and 5b [Qi et. al. (2017)], respectively.

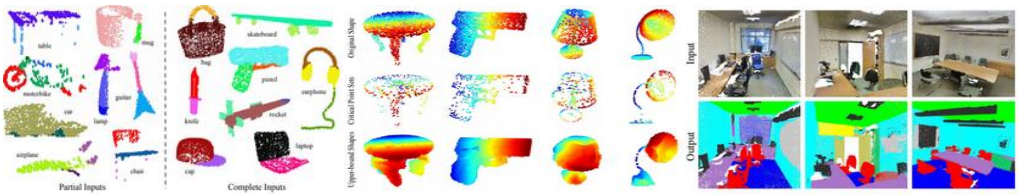


Image 5: a) Part Segmentation b) Critical points and Upper bound shape c) Semantic Segmentation

1.2. Indirect Methods:

Indirect methods aim to detect 3D line segments by detecting the edges in 2D images and then determine them in 3D space. In literature a plethora of such implementations and studies is proposed.

Lin Yangbin et. al. (2015) proposed an approach called Line-Segment-Half-Planes (LSHP) for 3D line segments extraction using plane intersection by applying it straight forward on the raw point cloud data. In general, it presents an analysis and implementation of LSHP on large point clouds, that are produced using a laser scanner in real-world scenarios. Firstly, the acquired point cloud is projected onto several image planes with a variety of observation points. Thus, a group of images which fully describe the region of a specific area, is produced. Then the LSD algorithm is implemented to detect 2D line segments in the 2D space. Further, the LSHP structure is used to eliminate the erroneous extractions and to set each line's geometric constraints. Afterwards, the detected points are projected in "V" shape [Lin Yangbin et. al. (2015)] back to the 3D space. Then, the "V" shape is decomposed into two non-parallel planes and the planes are detected i.e., their formula is defined. Finally, the line is produced via planes' intersection. The output of this implementation is depicted in Image 6 [Lin Yangbin et. al. (2015)].

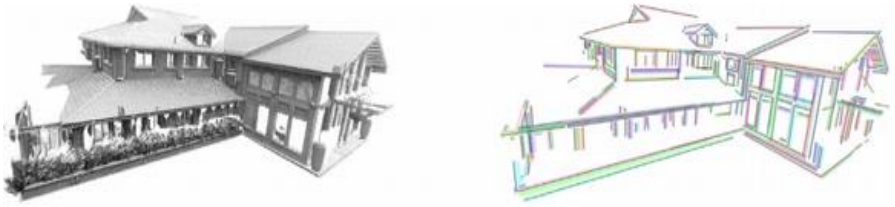


Image 6: Point cloud and the produced 3D lines

Xiaohu Lu et. al. (2019) presents a method for 3D line extraction using unorganized point clouds with millions points. Additionally, the proposed workflow was evaluated using known public datasets such as Bildstein 1 [Datasets], Castle [Datasets], St Sulpice [Datasets] and Facade [Datasets], with remarkable results. Firstly, the given point cloud is decomposed into smaller clusters, each cluster consists a plane. The pipeline that produces those clusters uses two main techniques: i) the region growing and ii) the region merging. Those steps aim to firstly demonstrate the small object's areas and then to identify those which in combination to each other will accurately produce one of each object's planes. Firstly, the small areas must be detected to extract their normal vector that is vital for the next steps. To achieve that, the KNN algorithm is applied directly to the unorganized point cloud. Then using each area's covariance matrix in combination with their eigenvalues and eigenvectors, a set of four values is associated with each 3D point. After that, a specific point is initialized to start the iterative procedure i.e., region growing and then each point with their four values, is traversed. If the traversed point satisfies three criteria is characterized as belonging to the region and the area expands. Each expanded area is associated with four numbers, which describe their characteristics, by implementing the normal vector calculation step again. Then two of the previous criteria are checked again. If they are satisfied the neighborhoods of the expanded areas are merged. This step is called region merging. Afterwards, one merged area is processed at a time by projecting its points onto an image-plane. The projection's x and y direction are defined using the area's normal vector while each point's coordinates are set using the area's central point. Additionally, a scale value is associated with each plane. To this end, the image pixels that correspond to a projected point are colored white i.e., 255, while the rest are colored black i.e., 0. Finally, 2D edge

detection algorithms are applied on every image produced and the detected 2D line segments are re-projected in 3D space. To achieve more accurate results a post processing pipeline is performed, in which potential outliers are removed and line segments that lie on the same line, are merged. Image 7 [Xiaohu Lu et. al. (2019)] illustrates the algorithm's results on the Castle and St Sulpice datasets.

Image 7: Results on Castle and St Sulpice datasets

Dolapsaki (2020) proposed an algorithm for edge detection, in unorganized point clouds, using one image. The proposed methodology was performed and evaluated in real-world data. Therefore, a laser scanning implementation in the ancient temple of Dimitra, in Naxos was implemented and the generated point cloud in combination with an SfM-MVS point cloud using Photoscan/Metashape software, were used. The point cloud that was produced from FARO 130 laser scanner was characterized by +/- 3-4 cm uncertainty. The final unorganized point cloud containing 737,767 points, was produced by acquiring points every five millimeters. Some of the photos which were used to produce the 3D point cloud, using the SfM-MVS pipeline, were selected for the tests. Firstly, an image is loaded. Then the user adds manually two points on the image, in order to choose the edge which will be detected in the unorganized point cloud. Of course this could be performed automatically, but for the sake of the tests it was not implemented at this stage.

Image 8: User's input

Afterwards, the line is determined via a least squares method, so as to detect it with the largest feasible accuracy. The following step is performing a rotation (ω, φ, κ) and a translation (dx, dy, dz) to the image coordinates according to image's extrinsic parameters. Furthermore, the RANSAC algorithm is implemented and the 3D line is fitted to the transformed edge points. For maximum accuracy, a relatively small threshold value is selected for the application of the RANSAC algorithm. After that, the parameters of the plane, defined from two of the edge's points and the exterior orientation of the camera, are calculated. Afterwards the Euclidean distance of each point from the plane is calculated, and the points

closer than a given value t (threshold), are stored as inliers while the rest are characterized as outliers. Then a data matrix that contains all the inlier points, which demonstrate the edge, in the world coordinate system, is constructed. All the outliers are removed, from the point cloud. The RANSAC algorithm is executed again to reduce the random noise, that is added to the data matrix, due to the edge's parallel lines. In conclusion, the detected edge points include all of RANSAC's inliers.

Image 9: Detected line presented by red color and the ground truth line presented by green color

Alshawabkeh, (2020) introduced a methodology for 3D line feature extraction by exploiting a combination of laser scanner and image-based techniques. The proposed workflow was applied to the ancient city of Petra in Jordan. The implementation aimed to efficiently detect cracks on the façade of the Treasury monument of Petra. Those cracks were produced by the running water due to an ancient construction of the Treasury's water system. Assuming that the LiDAR point cloud has already been produced, the region of interest i.e., the Treasury monument, is firstly captured from a diversity of viewpoints using a full-frame DSLR camera to achieve the most feasible accuracy. Thus, a sequence of RGB images is produced. Secondly, the produced LiDAR point cloud is converted into a set of depth images with the same characteristics as the RGB sequence. The depth maps are transformed to the resolution of the RGB images to exploit all the available information, from the RGB images. The depth maps are transformed using an interpolation to up-sample the depth images to the RGB resolution. In, fact the depth information remains the same because it is independent of the depth-image resolution. Then, the images and LiDAR products are transformed so as to refer to the same coordinate system by calculating the necessary translations and rotations. Afterwards, the Canny [Canny (1983) and (1986)] algorithm is performed on each RGB image and the linear features are revealed. Finally, the detected linear features are associated with the LiDAR data to reveal their 3D coordinates. The detected 3D features, on two images and the meshed laser model, are depicted in Image 10 [Alshawabkeh, (2020)].

Image 10: a) Detected Features on images b) Detected Features on LiDAR meshed model

Hofer et. al (2017) proposed an approach to efficiently detect and vectorize lines, to produce a line-based 3D model, by firstly detecting and matching those lines on images, which are taken from a diversity of

viewpoints, and then to accurately define them into 3D space. The approach presented as an alternative to the time-consuming MVS algorithm in a manner of a more simplified method which simultaneously provides accurately the 3D scene information. Assume that an unordered image sequence is given while each image pose is known due to an SfM implementation. Then an image edge detector is performed to define the image's lines. Hofer et. al (2017) make reference to two well-known approaches by Von Gioi, Jakubowicz, Morel & Randall (2008) and Akinlar & Topal (2011). Afterwards, the detected lines must be matched each other on the different images. To achieve that, firstly the given images are grouped to images with similar view i.e., *visual neighbors*, to reduce the computational need. Each of the detected lines on each image is characterized by two points i.e., *end points*. For every pair of lines segment in the two images (i, j) the epipolar line, of each of the image's i two end points, is specified onto image j. Then, the line, which lies on image-plane j, is determined and the intersections of it with the calculated epipolar lines are defined. Afterwards, each pair is associated with a score value and those with score larger than a pre-defined threshold value are stored as hypothetical good matches [Hofer, Wendel & Bischof (2013, February), Hofer, Wendel & Bischof (2013, September), Hofer, Maurer & Bischof (2014), Hofer, Maurer & Bischof (2015)]. Then, an evaluation step is performed to determine which of the hypothetical matches is the correct one by keeping only the matches with confidence formula higher than 1. The 3D hypothesis with the highest confidence value is chosen as the correct one [Hofer, Maurer & Bischof (2014)]. Finally, a graph-clustering algorithm is performed and produces a 3D line group which contains the 2D lines, the corresponding 3D line and some parts of the infinite line on which the produced 3D line lie. The 3D line is determined using depth estimations provided by the 2D residuals i.e., lines while the direction of the 3D line is calculated using the Singular Value Decomposition (SVD) technique. Finally, a bundle adjustment technique is provided optionally aiming to produce a more accurate 3D line-model. An example of the produced 3D model provided using the software *Line3D++* which is freely available and provided with Hofer et. al (2017) effort is depicted in Image 11.

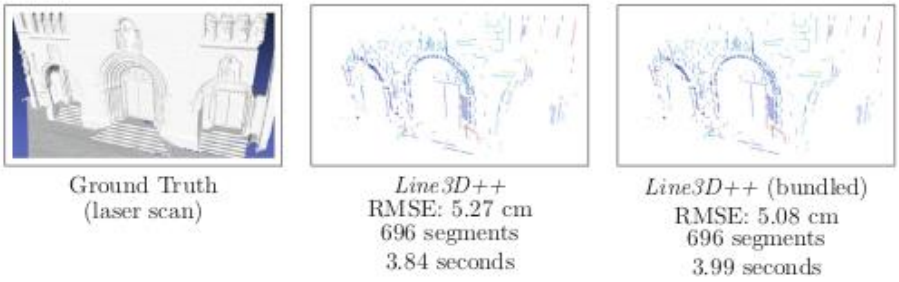


Image 11: Line3D++ results

Skentzos (2020) presented an automatic method to detect lines in unorganized point clouds. He proposed an improvement of the Canny [Canny (1983) and (1986)] algorithm by enhancing the used images using a bilateral filtering technique. Firstly, the proposed algorithm reads the images. Secondly, it reduces their resolution to reduce the computational cost. Thirdly, it applies image enhancement filters on each of the multiple images. Fourthly, it detects the lines on each image by applying the Canny [Canny (1983) and (1986)] algorithm and it then resizes them back to the original resolution. Afterwards, it overlaps the original images with the Canny's images using the image's red channel. Finally, it uses a typical SfM-MVS software to produce the dense cloud using the constructed images. Thus, the semantic information i.e., the edges pass through the SfM-MVS workflow and are displayed in the dense point cloud with red color. A triplet of the constructed images, is depicted in Image 12 [Skentzos (2020)], while the produced dense cloud is displayed in Image 13 [Skentzos (2020)].

Image 12: A triplet of images which are is used in SfM-MVS workflow

Image 13: The produced dense point cloud

Wang et. al. (2015) proposed an automatic methodology that exploits point cloud information i.e., each point's x, y and z coordinates, to decompose it into n-parts and then render them into 3D geometric models. They aim to use those models for simulation applications like fire case simulation and building's energy losses simulation. Firstly, the given unorganized point cloud is cleaned from the noise present and then it is downsized. Secondly, the given point cloud is decomposed into n-planes by implementing a region growing technique. Thirdly, for each of the detected planes, a boundary detection algorithm is performed to extract a line model that efficiently represents the detected plane. Finally, the line model is classified to building components such as windows, doors and roofs. The region growing technique exploits each point's normal vector and curvature value associated with pre-defined threshold values in order to decide if a point lies on the same plane with its neighbors or not. The boundary of each building's component is extracted using logical and geometrical rules on a projection of each segmented plane's points onto a 2D plane. For instance, the windows are detected due to their representation, as closed empty boxes, the walls as objects vertical to the ground, the doors as closed empty boxes close to the ground and the roof as an object placed over the walls. Additionally, a further categorization was performed –if possible- like common or glass door according to the availability of the door's panel or not. Then a concave algorithm is performed to produce the line boundaries. Afterwards, a geometric model is created and then it is edited to be a solid model i.e., without gaps. Finally, the proposed method, was implemented in several case studies, one of which is displayed in Image 14 [Wang et. al. (2015)].

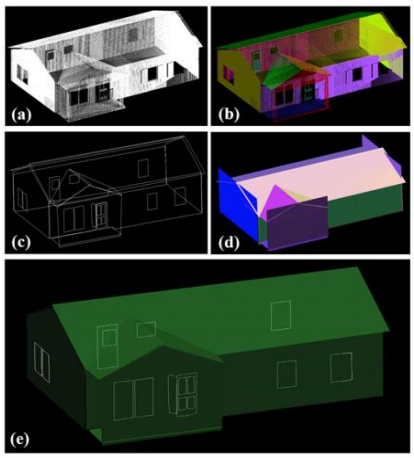


Image 14: a) Point Cloud b) Segmented Point Cloud c) Boundaries and Edges d) Edited rough model e) Created Semantic model

1.3. Commercial Software

PointCab [PointCab – Point Cloud Software] is a commercial program that is used for 3D measurement tasks, surfaces and volume calculations, mesh production, comparisons and profile creation while it is compatible with all the point cloud formats. There is a plethora of applications where PointCab can be a powerful and efficient solution, for instance architecture, heritage, surveying and road construction. Firstly, PointCab automatically processes the given point cloud and produces three views, the top, the front and the side view. Afterwards, it is easy to produce new views by using a section tool. The section tool can generate multiple sections of the given point cloud data, by setting a parallel offset and the quantity of the desirable sections. Furthermore, it contains an automatic workflow to produce 2D CAD plans, from point cloud data. That tool, named Vectorizer, takes as input a section, then it generates the 2D plan and finally saves the plan to a desired CAD format. The software's major steps to detect and vectorize the lines are: (i) Find Contours, (ii) Vectorize, (iii) Merge Lines, (iv) Insert Lines. Firstly, the vectorizer tool, identifies the lines and displays them on the given section. Then, the user can post-process the output in order to optimize the result, via the vectorizer's tool bar [PointCab – Point Cloud Software]. The provided vectorizer's tools are, "merge lines", "intersect lines", "draw lines freely" etc. while the toolbar is depicted in image fifteen. If the user presses the "shift" button it is possible to draw parallel lines and lines with 45^{o} angle.

Image 15: PointCab's vectorizer tools

Some experiments were carried out using the dataset Old Police Station in Monolithos, Rhodes. To take into account the object's complexity, the two parameters "Minimum line length" and "threshold" were redefined to 0.5 m and 162 respectively. The detected lines were not modified in order to optimize the result and are presented in Image 16, while the CAD export is displayed in Image 17.

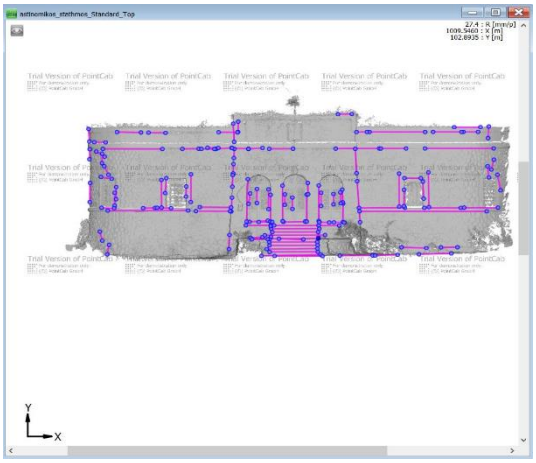


Image 16: Detected lines with 0.5m and 163 threshold value

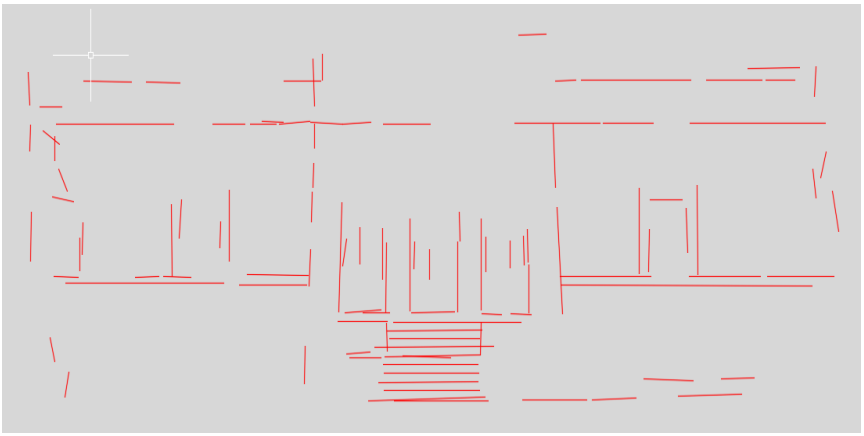


Image 17: Output in .dxf format with 0.5 m and 163 threshold value

Then a new similar experiment took place but with a maximum threshold value in order to achieve a more detailed result. Although the result i.e., Image 18 and Image 19 was more detailed, it was also noisier.

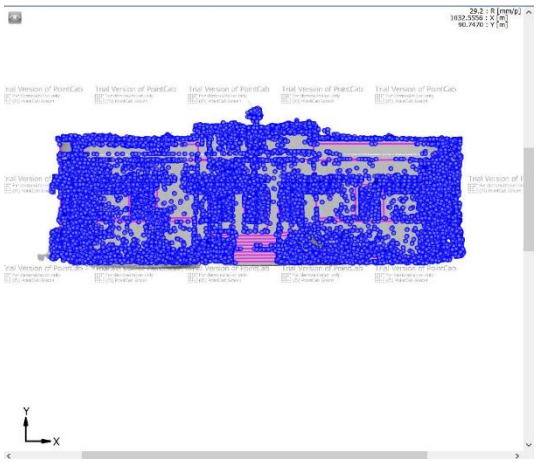


Image 18: Detected lines with maximum threshold value

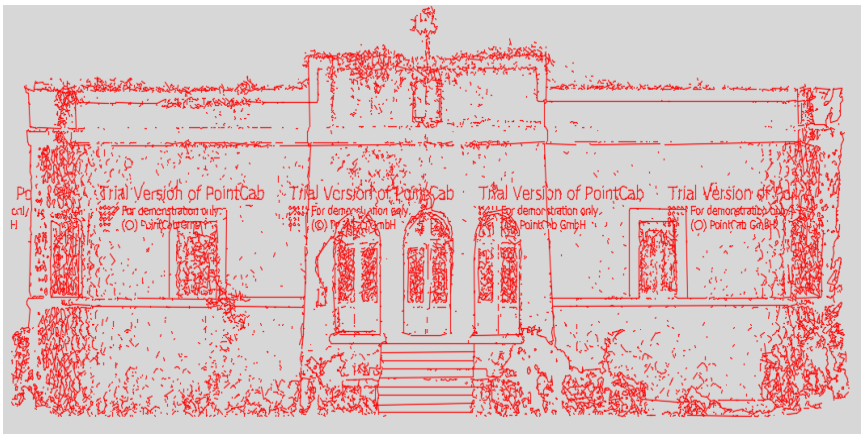


Image 19: Output in .dxf format with maximum threshold value

References:

Akinlar, C. and Topal, C., 2011, September. Edlines: Real-time line segment detection by edge drawing (ed). In 2011 18th IEEE International Conference on Image Processing (pp. 2837-2840). IEEE.

Alshawabkeh, Y., 2020. Linear feature extraction from point cloud using color information. *Heritage Science*, *8*(1), pp.1-13.

Bazazian, D., Casas, J. R., and Ruiz-Hidalgo, J. (2015, November), 'Fast and robust edge extraction in unorganized point clouds', *in 2015 international conference on digital image computing: techniques and applications (DICTA) (pp. 1-8). IEEE.*

Bienert, A., 2008, July. Vectorization, edge preserving smoothing and dimensioning of profiles in laser scanner point clouds. In *Proceedings of XXIst ISPRS Congress, Beijing, China* (Vol. 311).

Datasets [online] Available at: http://visionair.ge.imati.cnr.it/ontologies/shapes/ https://www.semantic3d.net/view_dbase.php?chl=1 (Accessed 10 February 2021)

Dolapsaki M. 2020 'Development of an algorithm for the detection of edges in point clouds using digital images' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (in Greek)

Donoser, M., 2013, September. Replicator Graph Clustering. In BMVC.

Felzenszwalb, P and Huttenlocher, D., (2004). Efficient graph-based image segmentation. Int. J. Comput. Vis 59 (2), 167–181.

Hofer. M., Maurer M and Bishof H. (2017) 'Efficient 3D scene abstraction using line segments' *Computer Vision and Image Understanding 157 (2017): 167-178.*

Hofer, M., Wendel, A. and Bischof, H., 2013, February. Line-based 3d reconstruction of wiry objects. In *18th Computer Vision Winter Workshop* (pp. 78-85).

Hofer, M., Wendel, A. and Bischof, H., 2013, September. Incremental Line-based 3D Reconstruction using Geometric Constraints. In BMVC.

Hofer, M., Maurer, M. and Bischof, H., 2014, December. Improving sparse 3D models for man-made environments using line-based 3D reconstruction. In 2014 2nd International Conference on 3D Vision (Vol. 1, pp. 535-542). IEEE.

Hofer, M., Maurer, M. and Bischof, H., 2015, October. Line3d: Efficient 3d scene abstraction for the built environment. In German Conference on Pattern Recognition (pp. 237-248). Springer, Cham.

Jain, A., Kurz, C., Thormaehlen, T and Seidel, H.-P., 2010. Exploiting global connectivity constraints for reconstruction of 3D line segments from images.

Lin Y, Wang C. Cheng J., Chen B., Jia F., Chen Z. and Li J (2014) 'Line segment extraction for large scale unorganized point clouds' *ISPRS Journal of Photogrammetry and Remote Sensing 102(2015) p. 172-183*

Lu, X., Liu, Y. and Li, K., 2019. Fast 3D line segment detection from unorganized point cloud. *arXiv preprint arXiv:1901.02532*.

Mitropoulou K. 2017 'Development of a procedure for the detection of planes and edges in unorganized point clouds' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (https://dspace.lib.ntua.gr/xmlui/handle/123456789/44820) (in Greek)

Mitropoulou, A., Georgopoulos, A., 2019. An Automated Process to Detect Edges in Unorganized Point Clouds, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-2/W6, 99-105, https://doi.org/10.5194/isprs-annals-IV-2-W6-99-2019, 2019 Skentzos O. (2020) 'Development of an automated algorithm for detecting edges in point clouds' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (https://dspace.lib.ntua.gr/xmlui/handle/123456789/51122), (in Greek)

PointCab – Point Cloud Software [Online]. Available at: https://www.pointcab-software.com/en/ (Accessed 25 November 2020)

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017) 'Pointnet: Deep learning on point sets for 3d classification and segmentation' *in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).*

Von Gioi, R.G., Jakubowicz, J., Morel, J.M. and Randall, G., 2008. LSD: A fast line segment detector with a false detection control. IEEE transactions on pattern analysis and machine intelligence, 32(4), pp.722-732.

Wang, C., Cho, Y. K., and Kim, C. (2015) 'Automatic BIM component extraction from point clouds of existing buildings for sustainability applications' *Automation in Construction, 56, 1-13.*

2. Theoretical Approach

2.1 Structure from Motion – Multi View Stereo

A fundamental problem in photogrammetry and computer vision is object 3D reconstruction, which aims to accurately determine the real shape, and color of objects and reconstruct them in 3D digital space. There are two ways that are used to manage the 3D reconstruction problem, the passive and the active one. On the one hand, passive methodologies, use data from sensors sensitive to a specific light spectrum, mostly the visible light (cameras) to reconstruct accurately the object of interest. In fact, passive visible light methodologies exploit the visible light spectrum by capturing the reflected radiance from the object's surface i.e., image production technique. Then the acquired sequence of images is used for scene understanding and finally for input in the 3D model production pipeline. On the other hand, active methodologies use techniques, that measure the distance of each acquired point by exploiting the speed of light and the beam flight time from the instrument to the object and vice versa. These techniques immediately estimate the distance of each point in space (depth), and consequently of the millions which are acquired. Widely known active instruments used in survey documentation tasks are laser scanners. Both methodologies are useful and necessary for similar or different case studies, while both are characterized by advantages and disadvantages. For instance, image-based approaches are collecting the real-world colors due to visible light usage, in comparison with laser scanner approaches that miss true color information in most of the cases. Laser scanner approaches could be used in low light conditions, while the image-based approach could not, without external illumination equipment e.g., studio lights. In this diploma thesis, image sequences are used for the 3D reconstruction task. In fact, the 3D reconstruction pipeline is separated to some major steps. In this effort, the 3D reconstruction workflow is applied until the dense cloud production i.e., the steps, from the dense cloud production to the 3D model creation, are omitted.

In general, the dense point cloud is created using Structure from Motion and Multi View Stereo algorithms. Firstly, images' poses i.e., relative translation and rotation matrices are estimated. The term 'relative' is used because the images' poses are estimated with respect to a reference image. In most cases, the first image in image sequence is chosen as reference image. Hence, the reference image's optical center position i.e., X, Y, Z, and camera's orientation angles i.e., ω, φ, κ equal to zero. Then for each subsequent image, the optical center position and camera orientation are determined via the position's and orientation's difference from the reference image. This step is performed by the standard SfM workflow. Necessary for this procedure is the determination of points on the images and their corresponding points on neighbouring images, which is realized by a suitable interest operator, like SIFT, and appropriate matching techniques. In parallel with the provided pose for each image the SfM workflow produces a sparse point cloud, which contains the determined homologue points. Structure from Motion algorithms simulate human and other living creatures' vision ability [Wikipédia, (2020) Structure from Motion]. To be more specific, as humans use two views i.e., left, and right eye, to acquire the scene's depth information, the SfM workflow uses image pairs or multiple combinations, to reconstruct pairs of 3D rays for depth information retrieval from the rays' intersection. In order to calculate the 3D coordinates with the most feasible accuracy, the 3D points are produced from the intersection of, preferably at least three 3D rays i.e., the point of interest is depicted at least in three images, and then an optimization methodology is implemented to determine the 3D scene points and cameras' poses.

Structure from Motion:

The standard SfM pipeline consists of three major steps, i) feature extraction for each individual image ii) feature matching for neighbouring images iii) parameter solving based on iterative bundle adjustment [Jiang S., Jiang C. & Jiang W. (2020)]. There are three SfM general approaches the incremental SfM,

the global SfM and the out-of-core SfM. In incremental SfM, camera poses are retrieved by firstly solving an initial pair and then by inserting one image at the time. In global SFM [Hartley & Sturm (1997), Triggs et. al. (2000)], the poses of all cameras are solved for at the same time [Wikipédia, (2020) Structure from Motion]. An intermediate approach is out-of-core SfM, where several partial reconstructions are computed and then are integrated into a global solution [Wikipédia, (2020) Structure from Motion]. In comparison with global SfM the incremental SfM pipeline, provides higher reconstruction precision due to the Bundle Adjustment (BA) technique which is applied [Jiang S., Jiang C. & Jiang W. (2020)]. Incremental SfM workflow utilizes iterative local BA and finally a global one, while the global SfM utilizes only a global BA technique [Jiang S., Jiang C. & Jiang W. (2020)]. A further analysis for the BA technique, is provided in the §2.1.4. In fact, incremental SfM workflow is the most commonly used approach.

Incremental SfM pipeline can be separated into two main parts, the correspondence search, and the incremental reconstruction as San Jiang et. al. mentioned. The correspondence search is divided into three steps, the feature extraction, the feature matching, and the geometric verification; while the incremental reconstruction can be divided into five steps, the initialization, for the best image pair selection; the image registration, for orientation; the triangulation, for 3D point calculation; the bundle adjustment, as optimization method; and outlier removal [Jiang S., Jiang C. & Jiang W. (2020)].

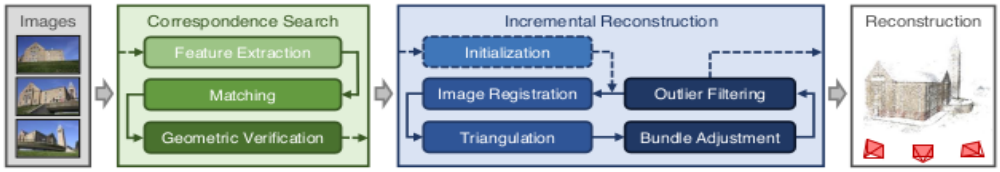


Figure 1: Incremental Structure-from-Motion Pipeline [Schönberger (2016)]

Correspondence Search:

1. Feature Extraction:

Feature extraction is a procedure that detects distinct areas in images. In fact, as distinct areas i.e., good features to track or good Key Points, are defined corners or blobs on the images [Jiang S., Jiang C. & Jiang W. (2020)]. Two edges in roughly orthogonal direction, determine a good corner while a blob is a region of an image in which some properties are constant or approximately constant and different from the background; all the points in a blob can be considered in some sense to be similar to each other [Wikipédia, (2020) Blob Detection]. The characteristics that define corners as distinct areas are, the invariance to translation, rotation, and illumination, and their different direction of x and y gradients. Every locally distinct location in an image is defined as Key Point. Every Key Point is associated with a descriptor vector which summarizes the local structure around it. There is a variety of corner feature extraction methods like Harris Corner [Harris & Stephens (1988)], Shi-Tomasi [Shi & Tomasi (1994), Tommasini et. al., (1998)], Förstner detectors [Förstner et. al. (2009)] and Difference of Gaussians [Lowe D. G (1999) & (2004)]. Firstly, for a 2D input i.e., (x, y) a local patch is computed. Then, for each local patch a 2x2 structure matrix is produced (Jacobian Matrix) which contains the gradient information. More precisely, it contains the derivatives of the image in a local patch while it accumulates the gradients in x and y direction. The structure matrix:

- Is the data structure which is used to detect the image's edges and corners.
- It accumulates the pixel's intensity changes in an image's local area.

Chris Harris & Mike Stephens (1998) presented the structure matrix as:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$X = I * [-1\ 0\ 1] \quad (2.1)$$

$$Y = I * [-1\ 0\ 1]^T \quad (2.2)$$

Where:

- I: image intensity values.

The eq.2.1 is almost similar with the derivative of image intensity values with respect to x direction i.e., $\frac{\partial I}{\partial x}$, while the equation eq.2.2 is almost similar with the derivative of image intensity values with respect to y direction i.e., $\frac{\partial I}{\partial y}$.

The elements of the structure matrix are:

$$A = X^2 * w \quad (2.3)$$
$$B = Y^2 * w \quad (2.4)$$
$$C = (XY) * w \quad (2.5)$$

Where:

- X and Y are the results of equation 2.1 and 2.2 respectively.
- $XY = (I * [-1, 0, 1])(I * [-1, 0, 1]^T)$
- w is the local patch around the potential Key Point.

Additionally, the structure matrix can also be defined using the Sharr or Sobel operator instead of the 1D arrays in eq.2.1 and eq.2.2. To be more specific:

$$X = (Dx * I)2 \quad (2.6)$$
$$Y = (Dy * I)2 \quad (2.7)$$
$$XY = (Dx * I)\,(Dy * I) \quad (2.8)$$

Where:

- Dx, Dy denotes the Sobel (Gao et. al. (2010), OpenCV, (2020) Sobel Derivatives, Jähne et. al. (1999), Wikipédia, (2020) Sobel Operator) or Sharr (Jähne et. al. (1999), Wikipédia, (2020) Sobel Operator x and y matrices. Those matrices are:
- Sobel:

$$D_y^{\text{Sobel}} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.9)$$

$$D_x^{\text{Sobel}} = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (2.10)$$

- Sharr:

$$D_y^{\text{Scharr}} = \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \quad (2.11)$$

$$D_x^{\text{Scharr}} = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad (2.12)$$

Further, it is possible to understand if a structure matrix refers to an edge or corner by comparing the structure's matrix elements i.e., A, B, C values. Specifically, if the structure's matrix first element i.e., (0, 0) or A and last i.e., (1, 1) or B are similarly large while the second i.e., (0, 1) or C and the third element i.e., (1, 0) or C are significantly small, then the matrix refers to a corner. The general idea is that a structure matrix refers to a corner if two values are similar and the other two are significantly small. If the structure matrix represents a line, it contains only one large value while the rest are significantly small. Finally, if the structure matrix contains only significantly small values the area is textureless e.g., white wall. Additionally, same conclusions can be derived from the eigenvalues and eigenvectors of the structure matrix. On the one hand, one large and one significantly small eigenvalue corresponds to edge structure matrix, because it means that the gradients of the local area are pointing more or less to the same direction. On the other hand, two similarly large eigenvalues correspond to a corner structure matrix, because gradients are pointing to different directions.

The disagreement between Harris Corner [Harris, C.G. and Stephens, M., 1988, August], Shi-Tomasi [J. Shi and C. Tomasi. 1994, J. Shi and C. Tomasi. 1994, Tommasini, T et. al., 1998] and Förstner detectors [Förstner, W. et. al., 2009] relies on the usage of different decision criteria for detecting a corner. Additionally, Förstner approach offers a sub-pixel corner estimation. More precisely, Harris Corner characterizes a region as corner by the calculation of a criterion R:

$$R = det(M) - k(trace(M))2 = \lambda1\lambda2 - k(\lambda1 + \lambda2)2 \quad (2.13)$$

Where:

- M is the Structure Matrix,
- λ1 and λ2: the eigenvalues of the Structure Matrix.
- k ε {0.04, 0.06}

When:

- $|R|$ = 0 (λ1 = λ2 = 0) → flat textureless region
- $R < 0$ ($\lambda1 >> \lambda2$ *or* $\lambda1 << \lambda2$) → edge
- $R >> 0$ ($\lambda1 = \lambda2$) → corner

Shi-Tomasi approach calculates the minimum eigenvalue and then compares it with a threshold value T. If the calculated eigenvalue is smaller than the threshold value T the structure matrix refers to corner. The provided formula in order to detect the smaller eigenvalue is:

$$\lambda min(M) = (\frac{trace(M)}{2}) - (\frac{1}{2})(\sqrt{((trace(M))2 - 4det(M))}) \quad (2.14)$$

Where:

- M is the structure matrix

When:

- $\lambda min(M) >= T$ → corner

Finally, the described approaches, in order to detect the corner's point location, search into the local region for the position with the maximum value of R or λmin. The detected position defines the corner's point location.

Apart from those algorithms, Difference of Gaussians (DoG) [Lowe D. G 1999, 2004] is a widely applied blob feature extraction technique. The convolution operation constitutes a vital contribution to DoG algorithm. In mathematics (in particular, functional analysis), convolution is a mathematical

operation on two functions (f and g) that produces a third function that expresses how the shape of one is modified by the other [Wikipédia, (2020) Convolution].

Firstly, a Gaussian Kernel with various sizes is applied to each image, producing a Gaussian pyramid for each of them. A Gaussian Kernel is a kernel, which produces a new smoothed image when it is convolved with the original image. In a Gaussian pyramid, subsequent images are weighted down using a Gaussian average (Gaussian blur) and scaled down [Wikipédia, (2020) Gaussian pyramid]. Each pixel containing a local average corresponds to a neighborhood pixel on a lower level of the pyramid [Wikipédia, (2020) Gaussian pyramid]. Then every Gaussian smoothed image, in the image's Gaussian pyramid, is subtracted from the others. This operation is called Difference-of-Gaussians. Specifically, DoG is a band-pass filter because it allows only the frequencies that lie between two differently low-pass filtered images i.e., the used smoothed images in the subtraction operation. Finally, it searches into the stack of images, which were produced via the Difference-of-Gaussians implementation, to find points that are locally distinguished in x, y or z direction, the x and y directions are referred to the subtracted image itself while the z direction is referred to each level above and beneath of image's Difference-of-Gaussian's level. DoG algorithm and SIFT feature descriptor were introduced by Lowe D. G. (1999) and Lowe D. G (2004). To conclude, every Key Point must be associated with a descriptor vector, which is the Key Point's "signature," and is later used in the feature matching task.

2. Feature matching:

The feature matching procedure aims to detect a point in every image in the image sequence on which is depicted. More precisely, for every detected Key Point, the feature matching procedure, detects its position on every image, in which it appears. Image matching is a fundamental task in computer vision applications like 3D reconstruction from multiple images, and scene - object recognition. Scale, orientation, and location invariant feature matching procedures are not handled efficiently by traditional corner detector approaches, mainly due to lack of relevant information about the provided Key Points i.e., descriptor vectors are not included. Thus, several feature descriptors were developed and proposed such as SIFT (Lowe 2004), SURF (Bay et. al. 2006), BRIEF (Calonder et. al. 2011), and ORB (Rublee et. al. 2011). These feature descriptors associate each Key Point with a descriptor vector. In fact, a descriptor vector is a representation of the point's local area and is used like the point's "signature" for the feature matching task. Additionally, feature matching can be considered as the problem of searching the nearest neighbor with the smallest Euclidean distance between two descriptor sets [Jiang et. al. (2020)] i.e., descriptors are used in order to compare key points and find which are similar (corresponding points). All these descriptor approaches differ in the descriptor's production methodology. Especially, SIFT and SURF provide non-binary while BRIEF and ORB provide binary descriptor vectors.

"Scale Invariant Feature Transform i.e., SIFT is an algorithm for extracting distinctive invariant features from images" [Lowe 2004]. Additionally, SIFT is invariant to translation, image rotation, and image scale [Lowe 2004]; while it is partially invariant to illumination changes, affine transformation, and 3D projections. SIFT feature descriptor is used for a variety of applications such as image matching, visual landmarks detection and object recognition. In this diploma thesis SIFT is used to tackle the image matching task. SIFT algorithm provides a robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination [Lowe 2004]. Additionally, features which are produced using SIFT algorithm, are highly distinctive in the sense that, every feature can be correctly matched with high probability against a large database of features from many images [Lowe 2004]. SIFT's major steps can be summarized as:

- Scale-space extrema detection: Using DoG methodology, it detects scale and orientation invariant interest points for each image in the given dataset.
- Key Point localization: Using a detailed model, it acquires the point's location and scale for each interest point.
- Orientation assignment: Local image gradient direction is computed, and one or more orientations are determined for each Key Point position. All future operations are performed on

image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformation [Lowe 2004].

- Key Point descriptor: In the area around each Key Point, the local image gradients are calculated at the selected scale. Further, they are transformed in a beneficial representation.

The scale-space-extrema detection implements the DoG algorithm that was previously described. The subtraction operation between the images, into the scale space, is implemented for each octave (Figure 2). Then Local-Extrema-Detection is performed and aims to detect points that are characterized by a significantly different (lower or higher) value from all of their neighbors. As each point's neighbors, are characterized the eight neighbors in the under-process image and nine neighbors in the scale above and beneath (Figure 3) [Lowe 2004].

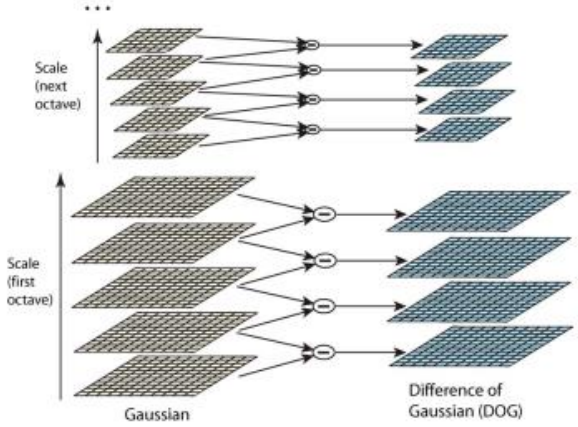


Figure 2: Gaussian-Scale-Space and Difference of Gaussian [Lowe 2004]

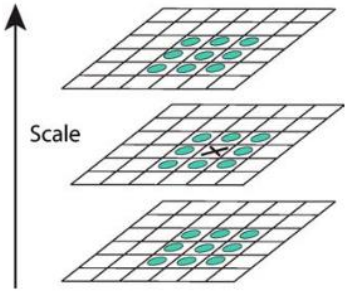


Figure 3: Maxima Minima Detection [Lowe 2004]

Then, the produced Key Points are detected with the most feasible accuracy using Brown and Lowe, 2002 methodology. Afterwards, each Key Point is associated with a consistent orientation in order to transform them into rotationally invariant. Firstly, an orientation histogram is produced using the gradient orientations of points which are randomly acquired around Key Point's area. Each histogram contains 36 classes while each addition into the class is associated with a weight value i.e., gradient magnitude, and with a Gaussian weighted circular window [Lowe 2004]. Finally, the three highest histogram values that are closest to the peak, are detected and a parabola is fitted to them in order to detect peak's position with a better accuracy.

To conclude, SIFT computes Key Points using DoG and descriptors using gradient histogram. Each Key Point's descriptor looks like Figure 4 [Lowe 2004]. These descriptors are used for the image matching task, even if the image database is significantly large. The major disadvantage of the SIFT algorithm is the computational cost. To handle real-time applications e.g., SLAM, BRIEF, ORB, and other binary descriptors were produced.

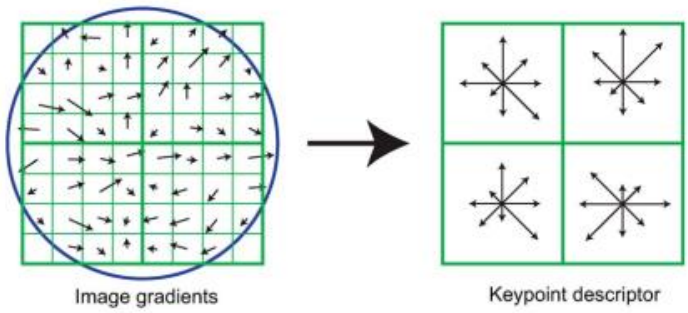


Figure 4: Image Gradients to Key Point descriptor [Lowe 2004]

Feature matching is performed by comparing each Key Point's descriptor with all the other descriptors. Lowe's ratio test is used to abort miss-matches. Firstly, the two closest descriptors, to the under-process descriptor (d0) are found (d1, d2) using Euclidean distance calculation. Then, the distance ratio is calculated. If the ratio value is more or less than one, the match is aborted due to the large ambiguity involved. If the ratio value is close to or even smaller than 0.5, the match is characterized as valid.

Binary descriptors methodology is fairly simplified compared to the one of SIFT. A brief description of their steps is:

- Select a patch around the under-process Key Point.
- Select a set of pixel pairs in that patch.
- Compare the intensity values for each pair.
- Concatenate them to a bit string.

The several binary descriptors approaches differ to the point sample methodology. In this manner, BRIEF proposed methodology, provides a rapid descriptor construction with a disadvantage of rotation variance i.e., if the camera is rotated a new descriptor is produced for the same Key Point. This disadvantage is handled by the ORB feature extractor. ORB is an extension of BRIEF, by adding rotation invariance. In addition, ORB uses FAST Key Point detector in combination with BRIEF feature descriptor. ORB algorithm is the most common for real-time applications e.g., SLAM.

3. Geometric Verification

Geometric Verification is a crucial task in epipolar geometry, aiming to distinguish the correct matches out of the detected ones due to the outlier's involved. Firstly, the obvious outliers are removed [Jiang et. al. (2020)]. Then the RANSAC method is implemented to detect the final matches by using an estimated transformation model. In order to achieve that, the Fundamental matrix is calculated using the seven-point algorithm [Zhang Z. (1997)] or the Essential matrix using the five-point algorithm [Nistér 2004]. The seven-point approach is used in the case of uncalibrated cameras while the five-point algorithm is used for calibrated cameras.

In general, due to the central projection transformation which is performed while an image is captured, in order to project the 3D world into the 2D image plane, some information is lost. Hence, from one image it is impossible to recover the entire structure of the 3D world [Hata & Savarese (2017)], leading to the use of at least two images for this task. In fact, three or more images are used to deal with this task, and accurately detect a 3D point. The geometry that relates the cameras, points in 3D, and the corresponding observations is referred to as the epipolar geometry of a stereo pair [Hata & Savarese (2017)]. The standard epipolar geometry set-up is illustrated in Figure 5 [Hata & Savarese (2017)].

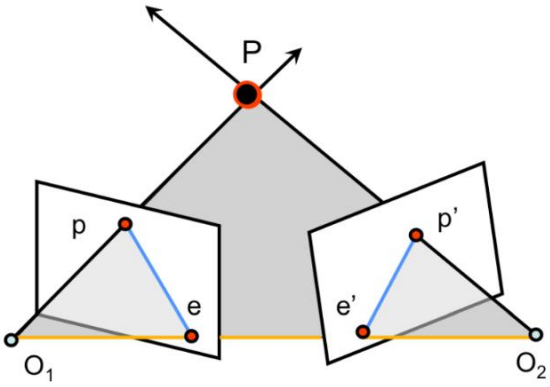


Figure 5: Standard epipolar geometry set-up

The above scheme illustrates camera's position using O_1, and O_2; and a set of corresponding points using p and p' respectively. The orange line segment, that connects the two camera perspective centers, is called baseline or epipolar-axis. P is defined as the point in 3D space imaged on p and p' in image space. The camera perspective centers, O_1, and O_2, and the 3D Point, P, define a plane called epipolar-plane. The intersection of the epipolar plane with the paleness of the first and second images, defines the epipolar-line containing point P, i.e., blue lines. The intersection of each epipolar-line with the baseline, define each image's epipole i.e., e and e'. In fact, exploiting, the knowledge of O_1, O_2, and P in one image, the epipolar plane can be defined. Then the second image's epipolar-line can be determined, by intersecting, its image-plane with the epipolar-plane. Hence, the corresponding point p' will be located on the second image's epipolar line. Thus, an understanding of epipolar geometry enables the construction of a strong constraint between image pairs without the previous knowledge of the 3D structure scene [Hata & Savarese (2017)].

The coordinate system used to implement the developed constraints to reconstruct a 3D scene, is a world coordinate system using O_1 i.e., first image's center, as point of origin. Then all the other image centers are determined using a rotation R, and a translation T, i.e., the offsets from the center of the first image. As mentioned above, the Essential or Fundamental matrix must be calculated to produce the transformation model that will be used by RANSAC algorithm in order to detect the presented erroneous matches.

Generally, Fundamental and Essential matrix, are used in stereo-geometry, to describe geometric relations between image pairs while contain all the information about the relative orientation. The Essential matrix is a 3 x 3 matrix, which has five degrees of freedom, rank 2 and is singular. Let $[a_x]b$ be the cross product between two matrices i.e., a, b. Then, Essential matrix calculations, using matrix multiplication, are [Hata & Savarese (2017)]:

$$([T_x]p')^T Rp = 0$$

$$P'^T[T_x]^T Rp = 0$$

$$P'^T[T_x]Rp = 0 \quad (2.15)$$

The matrix $E = [T_x]R$ is known as Essential matrix [Hata & Savarese (2017)]. The developed epipolar constraint is: $p'^T Ep = 0$. Essential matrix is applied in order to find the epipolar lines that are associated with p and p', for instance, the detection of $I' = Ep$, where I' is the epipolar line, on which the p' lies on and vice versa. In addition, the cross-product between the Essential matrix and an epipolar point i.e., e or e', equals to zero.

Furthermore, Fundamental matrix, is the general case of Essential matrix i.e., in the case of uncalibrated cameras. More precisely, Fundamental matrix encodes information about the camera matrices K, K' and the relative translation T and rotation R between the cameras [Hata & Savarese (2017)]. Similarly,

Fundamental matrix formula is $F = K'^{-T}[T_x]RK^{-1}$. The Fundamental matrix has seven degrees of freedom contrary to the Essential matrix which has five, while the usage is the same i.e., to find the epipolar-line of the correspondence point. To conclude, by using the Essential or the Fundamental matrix, geometric constraints can be produced between image pairs, without a previous knowledge of the 3D scene and with or without a calibrated camera. The Fundamental matrix can be computed using the eight-point Algorithm that was introduced by Longuet-Higgins in 1981 and extended by Hartleyin 1995 [Hata & Savarese (2017)].

Incremental Reconstruction:

1. Initialization

The initialization step aims to detect the most suitable pair of images, in order to perform the two-view reconstruction. In an incremental SfM workflow a pair with an acceptable intersection angle and an appropriate number of well-distributed features is selected as the most suitable pair [Jiang et. al. (2020)]. The initialization is a crucial step because a potentially miss-detected pair will destroy the 3D reconstruction pipeline. Additionally, the robustness of the 3D reconstruction, as well as its accuracy, and implementation are dependent on seed location of the incremental process [Schönberger (2016)]. In fact, the selection of an image with a variety of overlapping images i.e., increased redundancy, leads to a significantly improved robustness and accuracy in the reconstruction pipeline [Schönberger (2016)]. However, using an image associated with a few overlapping images results to the acceleration of the overall process. In general, the image matching task is the most time-consuming step in the overall workflow. Thus, the technique that is chosen to simplify the image matching procedure is a vital step in the 3D reconstruction workflow. Hence, several approaches have been introduced such as feature matching using an approximate nearest neighbor (ANN) algorithm [Beis and Lowe, (1997)], restricting the number of features extracted from one image [Wu, (2013)], pruning redundant images involved in reconstruction [Frahm et. al. 2010a, Li et. al. (2008)]. Among the stated methodologies the most efficient and robust one is the selection of image pairs before the implementation of the image matching task [Hata & Savarese (2017)]. This method is divided into three method-categories, prior knowledge-based methods, visual similarity-based methods, and match pair selection based on the analysis of image topological connection network (TCN) [Hata & Savarese (2017)].

2. Image Registration

The principal purpose of image registration is to estimate each image pose, using the previously established two-view model. In case that an uncalibrated camera is involved, its intrinsic parameters are calculated too. The general idea is to register new images into the calculated two-view model by solving Perspective-n-Point (PnP) problem. The already calculated poses are extended due to the pose of the newly registered image [Schönberger (2016)]. Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections on the image [Wikipédia, (2020) Perspective-n-Point]. The PnP solution is followed by RANSAC implementation to eliminate the produced outliers due to bias between 2D – 3D correspondences. The pose of the camera has six degrees-of-freedom (DoF), three for its rotation (ω, φ, κ) and three for its translation (x, y, z), with respect to the world coordinate system. PnP's formula is:

$$Sp_c = K[R[T]]p_w \quad (2.16)$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

"Perspective-n-Point model" [Wikipédia, (2020) Perspective-n-Point]

Where [Wikipédia, (2020) Perspective-n-Point]:

- $p_w = [x\ y\ z\ 1]^T$ is the world point in homogeneous coordinates.
- $P_c = [u\ v\ 1]^T$ is the corresponding image point in homogeneous coordinates.
- K is the matrix of the camera's intrinsic parameters.
    - γ is the skew factor
    - $f_x \rightarrow$ scaled focal length for x-axis
    - $f_y \rightarrow$ scaled focal length for y-axis
    - (u_0, υ_0) is the principal point position
    - S is a scale factor for the image point
- R is the desired 3D Rotation matrix.
- T is the desired 3D Translation matrix.

3. Triangulation

"Triangulation is the problem of detecting the point's location in 3D space given its location in two (or more) images captured with cameras with known calibration and pose" [Hartley, & Sturm (1997)]. This task is also referred to as ray intersection. To achieve that, two (or more) rays must be recovered while the 3D point is defined from the lines' intersection. In the most simplified case i.e., without noise, the solution is trivial [Hartley, & Sturm (1997)] but in real-world problems the triangulation is formulated as a least-squares minimization problem [Hartley, & Sturm (1997)]. Every registered image extends the produced scene by adding new information, i.e., new 3D points in the sparse point cloud. In addition, some of the existing 3D points are already observed from the added image due to its overlapped region with the previous one. Images that observe the same scene but from a different viewpoint can triangulate new 3D points. The term parallax is used to describe the apparent point's location changes in respect to a specified reference system, due to the observer's movement. Points, nearby to the observer, have greater parallax values contrary to the distant ones. The parallax is divided into the x-parallax and y-parallax components. In fact, x-parallax is related with the depth information i.e., z-axis; and is produced due to observer's movement, thus it constitutes a principal term for the scientific area of photogrammetry, while the y-parallax is eliminated during the process. Triangulation is an important step in SfM workflow, because it increases the stability of the existing model through redundancy [Schönberger (2016)] and allows the registration of new images by establishing new 2D – 3D correspondences. There are many algorithms for triangulation such as Poly [Hartley, & Sturm (1997)], Poly-Abs [Hartley, & Sturm (1997)], Mid-point [Hartley, & Sturm (1997)] and Linear-Eigen [Hartley, & Sturm (1997)], but their description does not fall within the scope of this effort.

4. Bundle Adjustment

Bundle Adjustment (BA) is the procedure for refining a visual reconstruction to produce jointly optimal structure and viewing parameter estimates [Triggs et. al. (2000)]. In fact, it is used to optimize both 3D points' and cameras' positions simultaneously -jointly- by minimizing the reprojection error using a function π -optimal-, that projects the 3D points back onto the image plane and a loss function p to potentially down-weight outliers [Schönberger (2016)].

$$E = \min\left(\sum_{k=1}^{n}\sum_{j=1}^{m} p_{kj}\left(\left\|\pi(P_c, X_k) - x_{kj}\right\|^2\right)\right) \quad (2.17)$$ [Jiang et. al. (2020)]

Where [Jiang et. al. (2020)]:

- P_c and X_k indicate a camera and a 3D point, respectively.
- $\pi(P_c, X_k)$ is the projection of point X_k on the camera P_c.
- x_{kj} denotes an observed image point.
- ||•|| denotes the L2-norm.
- p_{kj} is an indicator function.

Bundle Adjustment is named after the bunch of rays 'Bundle' from the image points to the points in 3D space, which are 'adjusted' optimally using both feature and camera positions simultaneously [Triggs et. al. (2000)]. Bundle Adjustment parameters are the 3D feature coordinates, camera poses and orientations, and calibrations. In fact, bundle adjustment is a fundamental theory of metrology while it is applied on various science fields such as photogrammetry, geodesy, and computer vision.

In an incremental SfM, an iterative implementation of Image Registration, Triangulation, and Bundle Adjustment steps is performed. Once the first pair is established by the Image Registration step, and triangulated, by the Triangulation step, it will be refined using the BA technique. A new image is registered and passed through the triangulation procedure. Then a local BA is performed to eliminate the collected noise due to the newly added image [Jiang et. al. (2020)]. In the end, after all images have been processed, a global BA is implemented to optimize both the camera poses and produced 3D points [Jiang et. al. (2020)].

The Bundle adjustment and similar adjustment computations are formulated as nonlinear least squares problems [Triggs et. al. (2000)]. Traditional BA techniques use quadratic cost functions to calculate reprojection errors while the robustness is provided by explicit outlier screening [Triggs et. al. (2000)]. The modern approaches use a non-quadratic M-estimator-like distributional models to provide a more generalized methodology than the traditional ones [Triggs et. al. (2000)].

There is a plethora of BA algorithms developed during the sixty years of research on this topic, like Simple methods [Jiang et. al. (2020)], Hierarchical methods [Jiang et. al. (2020)] and Global model constrained methods [Jiang et. al. (2020)], but the further analysis is out of the scope of this diploma thesis. A brief history of the main developments in bundle adjustment is provided by Triggs et. al. (2000). In fact, BA is one of the most crucial and demanding procedure in the SfM workflow, both to be understood and implemented.

5. Outlier Removal - Filtering

The filtering step is performed after each BA execution and it aims to eliminate the erroneous observations with respect to the model [Schönberger (2016)]. The criterion to eliminate an observation, is the reprojection error. More precisely, observations with large reprojection errors are eliminated. After that, for each recovered point the geometry, from which it was produced i.e., triangulation angle over all pairs of viewing rays, is checked [Schönberger (2016)]. Concretely, a minimum angle value is set as threshold, and every angle is compared with it. The rays that do not satisfy this value are omitted. Finally, while the global BA is performed, a further check to detect non-processed cameras is implemented [Schönberger (2016)].

Multi View Stereo:

In general, Multi View Stereo (MVS) algorithms construct the final 3D model, by using the SfM's output i.e., the camera poses and the sparce point cloud. The MVS algorithms can be classified into four categories, Voxel-based, deformable polygonal meshes, multiple depth map, and patch-based [Furukawa et. al. (2009)]. Another approach to classify MVS algorithms is by the dataset types that they can deal with. Concretely, those datasets can contain images of objects, scenes, and crowded scenes. An object's dataset is the simplest one and is characterized by a set of unordered images which are taken around of the object [Furukawa et. al. (2009)]. A scene dataset is characterized by images depicting the object of interest in addition with some noise i.e., trees, other unwanted objects, rocks [Furukawa et. al. (2009)]. A crowded scene dataset contains common images i.e., contains moving objects apart from the static structure of interest [Furukawa et. al. (2009)]. The last scene type consists of the most biased one and so, it is the most complicated scenario. According to the first categorization, multiple depth maps and patch-based approaches are able to efficiently process more challenging scene data sets. Two much appreciated MVS algorithms are CMVS-PMVS [Furukawa et. al. (2009) & (2010)] and DAISY [Tola et. al. (2012)]. The CMVS-PMVS algorithm is analyzed in the following.

The CMVS-PMVS (Clustering MVS, Patch-based MVS) approach was introduced by Furukawa et. al. 2009 & 2010. In fact, CMVS algorithm takes as an input the SfM output i.e., the camera poses and sparse point cloud, and transforms it into image clusters with practicable size. Furthermore, the sparse point cloud is considered from the CMVS algorithm as scanty examples of the dense reconstruction's points which will be produced using the PMVS algorithm. Besides the practicable cluster size construction, CMVS algorithm strives to produce clusters where each SfM point can be accurately determined by at least one of them [Furukawa et. al. (2010)]. The CMVS algorithm operates under three main constraints, compactness, size constraint and coverage [Furukawa et. al. (2010)]. The view clustering algorithm i.e., CMVS consists of four steps. The first two constitute a pre-processing step while the last two are repeated in a loop [Furukawa et. al. (2010)]. Figure 6 depicts the four steps of the view clustering algorithm [Furukawa et. al. (2010)].

Figure 6: Clustering View Algorithm step

1. SfM filter – merging SfM points [Furukawa et. al. (2010)]:

This step aims to decrease the number of the input SfM points in order to improve the execution time of the next three steps. Firstly, it starts by using a set of SfM points, then one random point is chosen and merged with its neighbors, the merged point is yielded while the merged and the under-process points are removed from the input set. The procedure is repeated until the input collection is vacant. Finally, the new input is the previously produced merged points and so on. When the process stops, the produced point cloud and viewpoints are a generalized version, that aims to enrich visibility information, of the input sparse point cloud and camera poses, respectively.

2. Image selection – removing redundant images [Furukawa et. al. (2010)]:

Firstly, the images are sorted with ascending resolution order if, of course, multiple resolutions exist. Then, each image is removed while the coverage constraint, of the remaining set of images, is checked. If the coverage constraint, still exists after the deletion, then the image is permanently removed. That step aims to remove the redundant information i.e., images, so as to reduce the execution time.

3. Cluster division – enforcing the size constraint [Furukawa et. al. (2010)]:

If a cluster does not satisfy the size constraint, it is split into smaller components [Furukawa et. al. (2010)]. This procedure is performed comparing an edge weight between an image pair that measure the pair's contribution to the MVS pipeline [Furukawa et. al. (2010)]. Those with the smallest scores are more likely to be excluded. The iterative procedure stops when the size constraint is satisfied for all the produced clusters.

4. Image addition – enforcing coverage [Furukawa et. al. (2010)]:

The cluster's division output may not satisfy the coverage constraint, so the fourth step reproduces the clusters by adding images so as not to disrupt the coverage constraint. Firstly, a list with possible actions is produced, where each of them measures the new cluster's effectiveness according to the viability of coverage constraint. Secondly, same actions are merged, and their effectiveness score is summed up.

Thirdly, the action's list is sorted in a descending order according to their effectiveness score. Finally, actions with score more than 0.7 times the highest score, are considered and the top action of this list is extracted simultaneously with all the conflicting ones.

Steps three and four are iteratively executed because each of them violates the others constraint. The process stops when both constraints i.e., size and coverage, are satisfied. Then PMVS algorithm is executed.

Some of the Multi View stereo algorithms, including PMVS, consist of three main steps. Firstly, the PMVS algorithm is executed and a dense point cloud is produced [Furukawa et. al. (2009)]. Secondly, the produced dense point cloud is converted into a polygonal mesh model [Furukawa et. al. (2009)]. Thirdly, the polygonal mesh model is refined using a polygonal mesh based MVS algorithm [Furukawa et. al. (2009)]. Since this diploma thesis, is not working on mesh models, polygonal mesh extraction and refinement are not further analyzed. The patch based MVS algorithm is divided into three steps, the initial feature matching, patch expansion, and patch filtering [Furukawa et. al. (2009)]. The first step aims to create a sparse set of patches while the expansion and filtering steps are iteratively executed to convert the patches into denser ones and remove miss-detected matches [Furukawa et. al. (2009)].

Furthermore, initial feature matching is divided into two steps, the feature detection and feature matching one [Furukawa et. al. (2009)]. The feature detection step detects blobs and corners on each image using the DoG and Harris Corner algorithms, respectively. To guarantee homogeneous coverage, a coarse regular grid of 32x32 pixels blocks, is overlayed on each image while each of the grid's block is used for the feature extraction task. The feature matching step uses the optical center of each image in combination with the detected features in order to produce several patches. More precisely, for each detected feature f, the set F of the same features f' (corners or blobs) are collected from the other images, if f' lies within two pixels from the corresponding epipolar lines. Then the 3D points that are related with the pair (f, f') are triangulated. Afterwards, these points are classified in ascending order in respect to their distance from the image's optical center. Finally, from each of the points, it endeavors to produce new patches [Furukawa et. al. (2009)].

Afterwards, the patch expansion step is implemented. This step aims to reconstruct at least one patch in every image cell and by repeating it to produce new so as to fill up the gaps. The patch expansion step is divided into two steps, identifying cells for expansion, and expansion procedure [Furukawa et. al. (2009)]. Given a patch p, its neighboring cells are extracted using some criteria [Furukawa et. al. (2009)]. Then the expansion procedure is implemented to produce a new patch for each collected image cell [Furukawa et. al. (2009)]. The whole iterative expansion procedure is presented in Furukawa Y. et. al. (2010) study. Finally, the filtering step is executed using a series of filters, in order to eliminate the wrongly produced patches [Furukawa et. al. (2009)].

References:

Bay, H., Tuytelaars, T. and Van Gool, L., 2006, May. Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.

Beis, J.S. and Lowe, D.G., 1997, June. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (pp. 1000-1006). IEEE.

Brown, M. and Lowe, D.G., 2002, September. Invariant Features from Interest Point Groups. In *BMVC* (Vol. 4).

Calonder M., Lepetit V., Strecha C., Fua P. (2010) 'BRIEF: Binary Robust Independent Elementary Features.' In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15561-1_56

Förstner, W., Dickscheid, T. and Schindler, F., 2009, September. Detecting interpretable and accurate scale-invariant keypoints. In *2009 IEEE 12th International Conference on Computer Vision* (pp. 2256-2263). IEEE.

Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., (2010a) 'Building Rome on a cloudless day'. *Springer, pp. 368–381*

Furukawa, Y., Curless, B., Seitz, S.M. and Szeliski, R., 2010, June. Towards internet-scale multi-view stereo. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 1434-1441). IEEE.

Furukawa, Y. and Ponce, J., 2009. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, *32*(8), pp.1362-1376.

Gao, W., Zhang, X., Yang, L. and Liu, H., 2010, July. An improved Sobel edge detection. In *2010 3rd International conference on computer science and information technology* (Vol. 5, pp. 67-71). IEEE.

Harris, C.G. and Stephens, M., 1988, August. A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, No. 50, pp. 10-5244).

Hartley, R.I. and Sturm, P., 1997. Triangulation. *Computer vision and image understanding*, *68*(2), pp.146-157.

Hata, K. and Savarese, S., 2017. CS231A Course Notes 3: Epipolar Geometry. *18c.)/K. Hata, S. Savarese.–2018–режим доступу: http://web. stanford. edu/class/cs231a/course_notes/03-epipolar-geometry. pdf.*

Jähne, B., Haussecker, H. and Geissler, P. eds., 1999. *Handbook of computer vision and applications* (p. 219 & p). San Diego: Academic press.

Jiang, S., Jiang, C. and 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.

[24] Li, X., Wu, C., Zach, C., Lazebnik, S. and Frahm, J.M., 2008, October. Modeling and recognition of landmark image collections using iconic scene graphs. In *European conference on computer vision* (pp. 427-440). Springer, Berlin, Heidelberg.

Lowe, D.G., 1999, September. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision* (Vol. 2, pp. 1150-1157). Ieee.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), pp.91-110.

Nistér, D., 2004. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, *26*(6), pp.756-770.

OpenCV, (2020) Sobel Derivatives [online]. Available at: https://docs.opencv.org/master/d2/d2c/tutorial_sobel_derivatives.html (Accessed: 20 December 2020)

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011, November. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). Ieee.

Schönberger JLFrahm, J.M., 2016, June. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA. Google Scholar Google Scholar Cross Ref Cross Ref*.

Shi, J., 1994, June. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 593-600). IEEE.

Tola, E., Strecha, C. and Fua, P., 2012. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, *23*(5), pp.903-920.

Tomasi, C. and Kanade, T., 1991. *Detection and tracking of point*. features. Technical Report CMU-CS-91-132, Carnegie, Mellon University.

Tommasini, T., Fusiello, A., Trucco, E. and Roberto, V., 1998, June. Making good features track better. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)* (pp. 178-183). IEEE.

Triggs B., Mclauchlan P., Hartley R., Fitzgibbon A 2000. Bundle Adjustment – A Modern Synthesis. *International Workshop on Vision Algorithms*, Sep 2000, Corfu, Greece. pp.298–372, 10.1007/3-540-44480-7_21. inria-00548290

Wikipédia, (2020) Structure from Motion [online]. Available at: https://en.wikipedia.org/wiki/Structure_from_motion (Accessed: 13 December 2020)

Wikipédia, (2020) 3D Reconstruction [online]. Available at: https://en.wikipedia.org/wiki/3D_reconstruction (Accessed: 13 December 2020)

Wikipédia, (2020) Convolution [online]. Available at: https://en.wikipedia.org/wiki/Convolution (Accessed: 13 December 2020)

Wikipédia, (2020) Gaussian pyramid [online]. Available at: https://en.wikipedia.org/wiki/Pyramid_(image_processing)#Gaussian_pyramid (Accessed: 13 December 2020)

Wikipédia, (2020) Blob Detection [online]. Available at: https://en.wikipedia.org/wiki/Blob_detection (Accessed: 13 December 2020)

Wikipédia, (2020) Perspective-n-Point [online]. Available at: https://en.wikipedia.org/wiki/Perspective-n-Point (Accessed: 13 December 2020)

Wikipédia, (2020) Bundle Adjustment [online]. Available at: https://en.wikipedia.org/wiki/Bundle_adjustment

Wikipédia, (2020) Sobel Operator [online]. Available at: https://en.wikipedia.org/wiki/Sobel_operator (Accessed in: 20 December 2020)

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

Zhang Z. (1997) 'Determining the Epipolar Geometry and its Uncertainty: A Review', *International Journal of Computer Vision* 27(2), 161–195 (1998)

2.2 Semantic Information

The term semantics (from the Greek word “σημαντικός”, significant) is used in several scientific fields such as linguistics, psychology, philosophy, and computer science [Wikipédia, (2020) Semantics]. Semantic theory is a part of the general theory of meaning [Speaks 2019]. Lewis (1976) distinguished the analyzing of meaning in two topics: “*first, the description of possible languages or grammars as abstract semantic systems whereby symbols are associated with aspects of the world; and, second, the description of the psychological and sociological facts whereby a particular one of these abstract semantic systems is the one used by a person or population.*”. This approach relates to the scientific fields of linguistics, psychology, philosophy, and sociology. In fact, those two categories can be abstracted into two questions: “*What is the meaning of this or that symbol?*” “*In virtue of what facts about that person or group does the symbol have that meaning?*” [Speaks 2019]. In linguistics, semantics subfield investigates the meaning of sentences, words, expressions, phrases etc. In addition, linguistics' approach to semantics theory, searches to find correspondences between the form and the meaning [Kroeger 2019]. Additionally, linguistics’ semantic theory is divided into five subfields, *“Formal semantics”*, *“Conceptual semantics”*, *“Cognitive semantics”*, *“Lexical semantics”*, *“Cross-cultural semantics”* and *“Computational semantics”* [Wikipédia, (2020) Semantics] but a further analysis is out of the scope of this diploma thesis. The approach of semantics in linguistics is strongly associated with Natural Language Process (NLP) in computer science. Moreover, semantics are used in programming languages such as python, C/C++, Java, PHP etc. and investigates the meaning of a potential line or expression etc., in a software (Nielson and Nielson 1992). Further, semantics in programming languages are divided into *“Operational semantics”*, *“Denotational semantics”* and *“Axiomatic semantics”* [Nielson & Nielson 1992].

In the fields of computer vision and image processing the term semantic information refers to an additional value, besides the RGB, that links each pixel or 3D point, with a meaning. In fact, the image, or the 3D point cloud, are decomposed into n-objects. Each object is associated with a value that refers to a meaning. More precisely, the input data, 2D or 3D, are segmented into n-clusters or groups i.e., set of pixels or 3D points, with the same characteristics and then each cluster is associated with a value which refers to a semantic information. The first step is called segmentation while the second one is called classification or semantic segmentation [Grilli 2019]. Segmentation and classification are demanding tasks while a variety of approaches have been developed through the years. In general, semantic segmentation techniques can be classified into traditional and recent approaches [Liu et. al. 2019]. Some segmentation techniques are “Region-Based Segmentation”, “Edge-Detection Segmentation”, “Segmentation based on clustering”, “Segmentation by model fitting”, “graph segmentation methods” etc. In this section, traditional segmentation techniques are mainly analyzed, however, some references are made to the most modern ones.

Region-Based Segmentation:

Yuheng and Hao (2017) divided the region-based approaches into threshold and region-growth segmentation. The simplest approach of image segmentation is called threshold, in which the pixels are grouped using a threshold value or values. The threshold techniques are separated into the local and the global methods [Yuheng and Hao 2017]. If a pixel has a value larger than T, is classified into a group while the rest are classified into a different one (Global method) e.g., a person from the background. If more than one threshold values are used i.e., n, the image will be divided into n+1, clusters (Local method). On the other hand, the region-growth approach, selects some seed points and by using a growth criterion, increasing the areas around the seed points while encapsulating the points with the same characteristics with, the under-process, seed point. In fact, the threshold approach is faster than the region growth method, due to the significantly lower usage of computational power but the region growth method extracts, most of the times, more accurate results. Instead, a major disadvantage of region growth

methodology is the dependency on the seed points. Kohler (1981) proposed a plethora of threshold segmentation approaches.

Grilli (2019) presented that, region-based segmentation techniques implement region-growing algorithms, which are divided into *"Bottom-up approaches"* and *"Top-down approaches"*. The first ones use initial points and then increase the regions around them using some constraints while the second ones firstly assume that all the points belong to the same class and then divide them into new clusters. The vital step of region growing algorithms is the selection of the region growing criterion either to 2D or 3D implementations. Liu et. al. (2018) presented a plethora of features, which are used in traditional segmentation methods, like SIFT [Lowe 1999 & 2004], SURF [Bay 2006], Harris Corner [Harris & Stephens 1988], Shi-Tomasi [Shi & Tomasi 1994], and FAST [Rosten & Drummond 2006]. The SIFT, Harris Corner and Shi-Tomasi, approaches are analyzed into SfM-MVS theoretical approach. Grilli (2019) presented the region-growing criterion in 2D images as pixel's color or intensity values while in 3D space e.g., 3D point cloud, besides the color, which can also be applied, proposed several geometric constraints such as surface orientation, curvature etc.

Let X denote an image i.e., a gird with values, P be a logical predicate defined on groups of connected pixels, and X_1, X_2, …, X_N be X's partitions. Zucker (1976) presented the conditions so that a partition to be a segmentation:

$$(i)\ X_1 + X_2 + \ldots + X_N = X$$

$$(ii)\ X_i\ i = 1, 2, \ldots, N \text{ is connected.}$$

$$(iii)\ P(X_i) = TRUE \text{ for } i = 1, 2, \ldots, N$$

$$(iv)\ P(X_i\ U\ X_j) = FALSE \text{ for } i \mathrel{!}= j, \text{ where } X_i \text{ and } X_j \text{ are adjacent.}$$

Finally, Zucker (1976) presented various approaches for region-growing segmentation but a further analysis of those techniques is out of the scope of this effort.

Kanade (1980) proposed an image model analysis in which the scene and the image domains are presented. This separation is proposed to distinguish the picture's domain features and the scene's domain features. For instance, two objects into the picture domain could be adjacent while in the scene domain it is not necessary for them to be next to each other. Further, Kanade (1980) proposed three types of knowledge, the *signal*, the *physical*, and the *semantic*, which can be used to deal with image segmentation tasks. Kanade (1980) defines the signal knowledge as "*The knowledge available at the signal level originates from the orderliness of the real world which is transformed by consistent rules into the signal"*. Additionally, he proposed three groups of segmentation techniques which exploit signal knowledge. These classes are, the local, the global, which are already analyzed, and their combination. Several approaches, for each class, are presented in his study. Finally, signal level knowledge provides results only for cues in picture domain, thus a further analysis to extract a meaningful information about the scene must be implemented. Hence, each segmented region must be associated with a name i.e., a class, which provides the essential meaning for them, such as wall, window, door etc. Kanade (1980) referred to those names as *"semantic names"* while he states that in order to define them a *"semantic knowledge"* is necessary. Finally, physical level provides the knowledge of the transformation between the picture and the scene domains. A paradigm of this kind of knowledge is the mapping between image's intensity values, and a surface with defined characteristics e.g., orientation, light direction etc.

Segmentation by model fitting:

These segmentation methods are often used in 3D data e.g., 3D point clouds, because they decompose the given data into primitive shapes such as spheres, planes etc. using a predefined model [Grilli 2019]. In addition, it is possible to classify the extracted segments to produce a meaningful information i.e., to perform a semantic segmentation procedure. In fact, these methods are fitting a model, which is defined by the minimum number of necessary parameters i.e., two for the line, three for the plane etc., onto the

given data for multiple candidate positions. Then, the positions, which satisfy most of the given data i.e., 3d points, are considered as results. Fischler and Bolles (1981) introduced the Random Sample Consensus algorithm (RANSAC). In comparison with the previously established techniques e.g., least squares, the RANSAC algorithm provides robustness when applied on noisy data. In fact, least squares fit a model e.g., a line, using all the given data. Hence, the outliers are also considered as points of the line and so the extracted line is depending on them. In contrast with the least squares, RANSAC is applied to the data, with a different approach. For instance, assume that the goal is to fit a line L, on a set of 3D points D. Firstly three parameters must be defined, (i) a threshold value t, which is the minimum distance between the under-process line and a 3D point, so as to be considered as line's point, (ii) a threshold value n, which defines the minimum number of points that a detected line must have to be considered as valid and (iii) the maximum iteration number for RANSAC to be implemented to detect the correct model. In fact, RANSAC algorithm classifies the given data into inliers and outliers, while simultaneously estimates the model's parameters. The RANSAC algorithm is used in several efforts such as by Mitropoulou & Georgopoulos (2019), Adam et. al. (2018), Canaz Sevgen & Karsli (2020) and Shen et. al. (2020). The RANSAC algorithm is also used in this diploma thesis in order to produce the final straight lines. Apart from RANSAC, Hough Transform (HT) is a well-known algorithm used in this kind of tasks. HT was introduced by Hough (1962) while several approaches have been implemented to improve the first version, such as the one introduced by Ballard (1981). Hough Transform is not used in this effort and so it is not further analyzed.

Segmentation based on clustering:

Two frequently used learning methods are the supervised and the unsupervised one. Nowadays, the progress of those fields is massive due to the increasing usage of Machine Learning (ML), Deep Learning (DL) and in general Artificial Intelligence (AI) techniques. In supervised learning the algorithms are enriched with previous knowledge for the data i.e., some examples, and exploiting it to perform the classification. For instance, assume that a satellite image must be classified into regions e.g., water, canopy, city, rocks etc. In supervised learning the user gives to the algorithm some examples of those categories i.e., small regions of pixels represent each of the desired classes. Then the algorithm exploits the previous knowledge, which was given by the user and classifies the entire image. On the other hand, unsupervised learning techniques are implemented directly on the given data without a previous knowledge. For the same example, the user defines the number of the segments e.g., 4, that the given data e.g., image, will be decomposed to. Then an algorithm is performed and returns the segmented image, in which each pixel is associated with a class. It is worth noticing that this procedure returns a segmented image, but those segments are not associated with a semantic information i.e., a meaning. Thus, a further process should be performed, to enrich them with a semantic information. A commonly used, unsupervised algorithm, is k-means which was introduced by MacQueen (1967). Firstly, the desired number of classes, in which the given data will be classified, is defined. Let n be this number. Secondly, n random samples, from the given data, are selected e.g., n points. Thirdly, for all the other samples a distance between them and the n randomly selected points, is calculated. Each point is grouped to the cluster with the minimum distance. Hence, n clusters are produced, containing some points. Fourthly, for each class the mean value is calculated, and the previous steps are executed again, using the n new samples i.e., the mean values. The procedure is iteratively implemented until the difference between the new mean values and the previous ones is significantly small. Although the k-means algorithm is a commonly used clustering algorithm, in this diploma thesis the DBSCAN is used, which was introduced by Ester et. al. (1996) while the generalized DBSCAN (GDBSCAN) was proposed by Sander et. al. (1998). DBSCAN's key idea is that each point, which is included in a cluster, has a neighborhood, containing more than a minimum number of points (Ester et al. 1996). This minimum value is called minimum points criterion, in this section. The terms *core points*, *border points* and *density-connected points* are further analyzed. More concretely, *core points* are the points inside a cluster while the *border points* lie on the boundary of each class. To define the *density-connected points* the terms *directly density-reachable* and *density-reachable* points must firstly be defined. A point p is *directly*

density-reachable from a point q if the point p is in q's neighborhood and if this neighborhood satisfies the minimum points criterion [Ester et al. 1996 *Definition 2*]. Furthermore, a point p is *density-reachable* from a point q if they are included in a sequence of i points, in which the i and i+1, are directly-density [Ester et al. 1996 *Definition 3].* Finally, *density-connected* are two points which have a third point from which both are *density-reachable* [Ester et al. 1996 *Definition 4]*. The major steps of DBSCAN algorithm are (i) Traverse the e (eps) neighborhood for each point in the given dataset and extract the *core points* which satisfy the *minimum points criterion* (ii) Find the *density-connected* points of *core-points* while the other points i.e., *border points* and noise are ignored (iii) If an ignored point can be associated with a nearby cluster it is included to it *(border point*), otherwise it is characterized as noise. In this diploma thesis the DBSCAN algorithm is used to decompose a given set of 3D points into sets of points, which lie on the same line.

Edge-Detection Segmentation:

A different approach to decompose an image into objects, is the segmentation based on edge-detection techniques. In contrast with other techniques such as region-based, clustering and model-fitting, which aim to directly extract the segmented regions, edge-detection segmentation aims to detect the borders between them i.e., the edges. The edge detection technique is a traditional segmentation approach [Grilli 2019]. Image edges are defined as sets of pixels at which, (i) the illumination is changed drastically i.e., *discontinuities (step edges),* or (ii) a *local extreme* is presented (line edges) or where (iii) *at least two edges meet each other (junction)* [Ziou & Tabbone 1998]. There is a plethora of techniques in the literature, which can be performed, to extract edges on images. Davis (1975) defined the edges as "T*he boundaries between two regions of different constant gray level"*. In unreal conditions i.e., in conditions without noise, a c*ross section of an edge orthogonal to its direction*, is presented as an instantaneous step Figure 1 [Davis 1975] where the x axis presents the image's x direction and the y axis presents pixel's intensity value i.e., image's grey level. Instead, due to the noise, the Figure 1 is an ideally situation, so in the Figure 2 [Davis 1975] a more realistic one is depicted. It is worth noticing that the angle between the edge's *ramp* and the x axis, is associated with the edge's sharpness [Davis 1975], and so it constitutes a valuable information to detect valid edges.

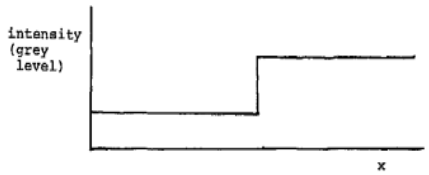


Figure 1: Step edge

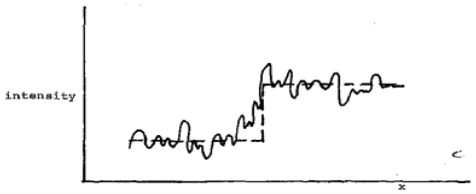
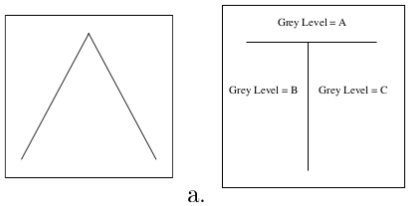


Figure 2: Noise step edge

The step edges occur where discontinuities are present i.e., where the illumination is changed drastically [Ziou & Tabbone 1998]. In order to detect the step edges (Figure 1 and 2) the first or second derivative must be calculated. Hence, if a *positive maximum* or a *negative minimum* is present in the first derivative a part of the step edge is localized there [Ziou & Tabbone 1998]. Instead, by estimating the image's second derivative, the edges are localized where a zero crossing is present. The step edge is the most common edge type. Apart from them, some others also occur such as the "*roof edges"* and the "*spike edge"* (Davis 1975). The "*roof edges"* are called *"lines"* by Ziou & Tabbone (1998) while they describe one more edge type the "*T-Junction"*. In fact, the term edges, encapsulates all the described categories. In Figure 3a the *"roof edges" or "lines"* are depicted while Figure 3b displays the "*T-Junction"* [Ziou & Tabbone 1998].



a. b.

Figure 3: a) roof edge or Line profile b) T-Junction

The *"roof edges"* or *"lines"* are defined by Ziou & Tabbone (1998) as the "*result from mutual illumination between objects that are in contact or from thin objects placed against a background"* while the "*T-Junction*" edges are produced when an edge obstructs another edge. *T-Junctions* are highly associated with the corners, due to their geometry. In fact, the edge detection and corner detection, techniques present similarities.

Edge extraction is a fundamental problem of the image processing field, which has aroused the interest of a vast number of scientists. Image edges provide a valuable information for a variety of applications in image processing as well as in computer vision. In fact, the edges can be exploited to understand the depth changes in an image or the changes of the orientation, reflectance, and shape of an object, which is depicted in it. Hence, edges are a useful information to understand the depicted scene. Several edge detection techniques have been produced to take advantage the edge's information. In the literature various edge detection operators are presented, including techniques and algorithms such as Roberts [L. G. Roberts 1965], Sobel [Sobel & Feldman (1968), Ziou & Tabbone (1998)], Prewitt [Prewitt 1970], Scharr [Kroon 2009], Kirsh [Kirsch 1971], Robinson [Wikipédia, (2021) Robinson Compass], Marr-Hildreth [Marr and Hildreth 1980] and Canny [Canny (1983) and (1986)].

Ziou and Tabbone (1998) divided the edge detectors into two categories the *"autonomous"* and *"contextual"*. The *"autonomous"* edge detectors do not use a-priori knowledge in order to detect the edges in contrast with the *"contextual"* ones. It is worth noticing that "*autonomous"* edge detectors are the most common ones while some of them are described in this diploma thesis. The "*contextual"* ones are out of the scope of this effort and so they are not further analyzed, but several algorithms can be found in Davis (1975), Matsuyama (1989) and Becerikli & Karan (2005). Despite the usage, or not, of a-priori knowledge, there are three main steps which edge detectors perform, (i) *differentiation*, (ii) *smoothing* and (iii) *labelling.* The differentiation evaluates the image's derivatives i.e. searches to find the amount of the gray level difference between a left and right or an upper and lower set of regions. Smoothing reduces the image's noise and the labelling eliminates the erroneous edges to increase the *signal-to-noise* ratio [Ziou & Tabbone 1998]. The three steps are working in combination with each other due to their dependency. For example, if the image is over smoothed, then it loses the desirable information, and so the labelling becomes a difficult process, without successful results. A further analysis of the three steps, is presented in Ziou & Tabbone 1998 study.

The first efforts, which were conducted to produce edge detection operators included only the differentiation step. Robert's, Sobel's and Prewitt's operators are some of the first approaches dealing with the edge detection task. Sobel's and Prewitt's operators are based on the following 3x3 masks to calculate image's derivatives.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -a & 0 & -a \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -a & -1 \\ 0 & 0 & 0 \\ 1 & a & 1 \end{bmatrix}$$

Where α is a different integer number for each case i.e., 1 (Prewitt) and 2 (Sobel) [Ziou & Tabbone 1998]. Another approach of Prewitt mask is the replacement of each 1 element with 1/3 in G_x and G_y (Skimage documents, (2021)). Edges are detected, according to the x and y direction, using the G_x, and

G_y kernels respectively. Thus, depending on edges' direction, x edges could be presented with white color, while the y with black. Edge's magnitude is calculate using the eq.2.18. Apart from the edge's magnitude, the edge's orientation is calculated using the eq.2.19. Edge's magnitude is used as the final criterion for the edge detection task using Sobel and Prewitt kernels.

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (2.18)$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (2.19)$$

The Sobel operator is also analyzed in the SfM-MVS theoretical approach, but with an additional smoothing implementation. In SfM-MVS the Scharr operator (Kroon 2009) is also analyzed. Scharr's kernels are:

$$D_x = \begin{bmatrix} \mathbf{3} & \mathbf{0} & \mathbf{-3} \\ \mathbf{10} & \mathbf{0} & \mathbf{-10} \\ \mathbf{3} & \mathbf{0} & \mathbf{-3} \end{bmatrix} \quad D_y = \begin{bmatrix} \mathbf{3} & \mathbf{10} & \mathbf{3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-3} & \mathbf{-10} & \mathbf{-3} \end{bmatrix}$$

Scharr operator was introduced, to improve the rotation invariance of the derivative kernels (Kroon 2009). In addition, Kroon (2009) provided a further analysis of the image's first and second order derivatives as well as the Hessian Matrix, while proposing an optimized kernel, called *FMINLBFGS*, which minimizes the pixel's angular error. Robert's magnitude is calculated using eq.2.18, while the x, and y kernels; and the orientation are:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\theta\,(x,y) = \arctan\left(\left(\frac{G_y(x,y)}{G_x(x,y)}\right) - \frac{3\pi}{4}\right) \quad (2.20)$$

Robert detector is more sensitive to noise, in comparison with Sobel and Prewitt, due to its smaller size. In general, before the implementation of Sobel and Prewitt edge detectors, a blurring is performed e.g., Gaussian Blur, in order to filter high frequency regions i.e., to reduce noise. Rosenfeld & Thurston (1971) proposed the approach of image smoothing in combination with the edge detection techniques. The proposed approach improved the edge detection results, because the image is firstly blurred, to reduce image's noise, and then the edges are extracted.

Apart from the methods, which use first order derivative to extract the edges, further methods can be found in the literature which use the second order derivatives, like the Marr & Hildreth (1980) approach, in which the Laplacian of Gaussian (LoG) method is implemented. The major steps of LoG algorithm are (i) Smoothing (ii) Edge enhancement and (iii) Zero-crossing detection. The smoothing step is preformed using the Gaussian blur kernel while the edge enhancement using the Laplacian operator, thus the method is called Laplacian of Gaussian. The Laplacian kernel (L) has several approaches such as:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

Laplacian [Ziou & Tabbone 1998] Laplacian (Skentzos 2020) Laplacian (Skentzos 2020)

Finally, the zero-crossing detection is performed by searching for zero values produced from the calculation:

$\nabla G_{(x,y)} * I_{(x,y)}$ (2.21) (Marr and Hildreth 1980). Where: ∇^2 is the *Laplacian*, $G_{(x,y)}$ is *a 2D Gaussian distribution* and $I_{(x,y)}$ is the given image. Kirsh and Robinson operators deal with the edge detection problem using different kernels for each direction i.e. N, NW, W, SW, S, SE, E and NE. For each direction, an edge-map is produced. Finally, these edge-maps are combined into one image. These operators are out of the scope of this diploma thesis and so they are not further analyzed.

The most common edge detection approach is the canny algorithm. In this diploma thesis the canny algorithm is performed to enrich the given dataset, with edge semantic information. The Canny algorithm [Canny (1983) and (1986)] was introduced in order to improve the edge detection techniques. In fact, it is a step further from the first order derivative techniques such as Sobel, Robert and Prewitt. More precisely, the Canny algorithm takes as input the output of the first order derivative methods and exploits it in order to improve the detected edges. The major steps of Canny algorithm are (i) Smoothing (ii) Compute Gradients (iii) Non-Maximum Suppression (iv) Hysteresis Thresholding. In Figure4 the major steps of the Canny algorithm are depicted.

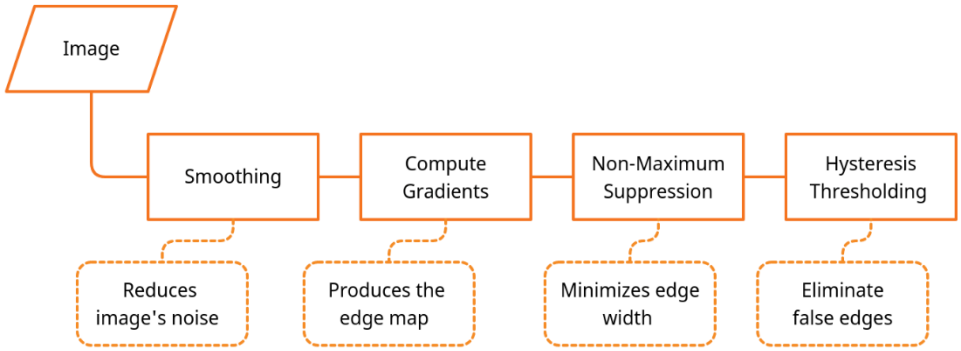


Figure 4: Canny pipeline

Firstly, the under-process image is filtered using a low-pass filter e.g. Gaussian blur. The result is a new image in which the previously high frequency regions have been smoothed. Thus, the presented noise has been decreased. The produced image is more suitable, than the original, because the edge detectors, which will be used in the image's gradient calculation, are sensitive to noise. When the noise has been removed, image gradients must be calculated. Hence, an edge detector such as Sobel, Robert or Prewitt, is applied. Using one of the available detectors, an edge map is extracted. Up to this step, the pipeline is the same as the previously described simple operators. The output i.e., the edge map, is used as an input to the Non-Maximum suppression (Kitchen and Rosenfeld 1982) step which aims to minimize the edge's width to 1 pixel (Rosenfeld and Kak 1976). To achieve that, the algorithm investigates if a pixel is a local maximum in its neighborhood as well as if the hypothetical edge is characterized with continuity along its direction i.e., θ. The direction is in a discreet space i.e. 2D image space, and so the θ values are classified into four categories according to the available directions (Figure 5) (Maini, and Aggarwal 2009). Then the direction is used to check the continuity constraint.

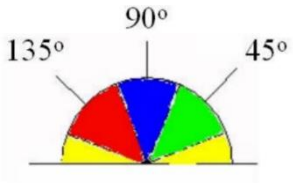


Figure 5: θ direction classes

Finally, the Hysteresis Thresholding step, in which the erroneous edges are eliminated, is implemented. To distinguish the valid from the non-valid edges, two threshold values are defined, one maximum and

one minimum. The edges with magnitude over the maximum value are automatically characterized as valid. The hypothetical edges with value smaller than the minimum one, are automatically excluded. The hypothetical edges, with magnitude between the two threshold values, are characterized as valid, if connected with a power pixel i.e., a pixel which is already characterized as edge, otherwise they are eliminated. In fact, the maximum and minimum threshold values are strongly associated with each individual image. Thus, even if the same object appears in both images, threshold values might be different between them. Usually, to define the minimum and the maximum values several trials are performed. In this diploma thesis, a live Canny implementation, inspired by Arapellis (2020), is used.

Cultural Heritage monuments and Semantic Information

The previously described methods have several applications in cultural heritage (CH) monuments. Semantic segmentation methods are frequently used in the conservation process of CH monuments. For instance, the semantic segmentation of a point cloud, which is acquired by scanning or photographing a Doric temple, into point sets of columns, capitals, stylobates etc. Semantic segmentation techniques are not only used in CH building environments such as temples, but also in artefacts like statuettes. There are various classification techniques and applications, which are implemented to CH building environments and artefacts. The classification approaches can be categorized into texture based and geometry based [Grilli 2019].

Both texture and geometry-based approaches exploit, image's or 3D point cloud's features, frequently in combination with machine learning techniques, in order to classify the 3D data, into objects. Grilli (2019) presented that, texture-based approaches firstly generate the 3D model using the given 3D point cloud. Secondly, orthophotos are produced and classified. Finally, the 2D classification results are reprojected into 3D space, i.e., the point cloud. The texture is exploited into the second step in which several image features are extracted e.g., the edges. Grilli (2019) presented several texture-based applications in CH monuments (i) "*Pecile wall in villa Adriana, Tivoli"* (ii) *"Sarcophagus of the spouses"* (iii) "*Bartoccini's tomb, Tarquinia"* and (iv) "*Porticoes in Bologna"*. The "*Pecile wall in villa Adriana, Tivoli"* case study implementation, aimed to recognize the *original* and the *restored*, wall regions while the wall was classified in respect to the opus construction technique (Grilli et. al. 2018). The *"Sarcophagus of the spouses"* case study implementation, aimed to recognize the *surface anomalies* and to measure the amount of the *cement*, which was used to *assemble* it. The *"Bartoccini's tomb, Tarquinia"* case study application aimed to identify the *deteriorated* regions of the walls. Finally, the "*Porticoes in Bologna"* case study application, was aimed to classify a small part of the porticoes into its *principal parts* and *architectural elements* such as doors, walls, windows etc.

The geometry-based approaches exploit, the point cloud's geometrical characteristics in order to decompose it into segments and then enrich them with the semantic information. Blomley et. al. (2014) presented various features, which can be used to semantically segment a given point cloud, such as the covariance features. The covariance features are extracted using the covariance matrix. More precisely, characteristics such as linearity, planarity, sphericity etc. are defined exploiting the eigenvalues of the covariance matrix. Geometry based approaches are very useful when visual information is not presented e.g., in point clouds from laser scanners. Grilli (2019) presented the major steps of the geometry-based approaches, using supervised machine learning algorithms, as (i) "*Manual annotation"*, (ii) "*Feature extraction"*, (iii) "*Model Training"*, (iv) *"Prediction"*, and (v) *"Validation"* but a further analysis of them is out of the scope of this diploma thesis. It is worth noticing that geometry-based approaches were implemented in "*Basilica Paestum"*, *"Temple of Neptune Paestum"* and "Mausoleum of Cesare Battisti in Trento, Italy" monuments but also, in "*Bartoccini's tomb, Tarquinia"* and "*Porticoes in Bologna"* monuments as the texture-based one [Grilli 2019 and Grilli et. al. 2019]. These experiments aimed to decompose the given data into specifically semantic segments. Murtiyoso & Grussenmeyer (2019) proposed an automated method in which a given historical building point cloud is semantically segmented into its architectural elements, using *geometric rules.* In fact, the proposed method is described using an example dataset, which is called Kasepuhan and presents the Kasepuhan Palace in

Cirebon, Indonesia. Firstly, the entire point cloud was divided into two classes the "*Attic"* and *"Body"* one. To achieve that, various horizontal planes were derived, and then various geometrical rules were exploited in order to segment the given point cloud. Then a region-growing segmentation technique based on Euclidean distance, was performed onto the building's body 2D cross-sections. Afterwards, the clusters' *circularity* was calculated, to distinguish them into *circular* and *non-circular* cross sections. The classified clusters were extended back into the 3D space. The entire segmentation was produced, using a method which separates the given point cloud using the produced segmented clusters. Finally, the floor was removed using the RANSAC algorithm and a new region-based segmentation was performed to eliminate the presented noise. The proposed method was evaluated into three different datasets.

Nguatem et. al. (2014) proposed a method, which detects the outlines of windows and doors in 3D point clouds of facades. The major steps of this algorithm are summarized as (i) "*Facade Segmentation"* (ii) *"Window and Door Localization"* and (iii) *"Model Selection*". The input data to this pipeline is a 3D point cloud produced either using the LiDAR technology or SfM-MVS workflow using images. The "*Facade Segmentation"* process firstly defines a plane parallel to the façade i.e., a vertical plane. Nguatem et. al. (2014) presented the difficulties to extract the appropriate information i.e., the boundaries of windows and doors, from just on plane e.g., due to construction or architectural imperfections although they further improved their technique. More precisely, they produced additional planes slightly parallel to the initial plane i.e., with a small angular difference along its normal direction. Thus, a set of slightly parallel planes, to the vertical one, is produced. Finally, the boundary points are extracted from those segmented planes and constitute a hypothesis of the real boundaries of the windows and the doors. The 3D detected points are projected on a 2D plane and the final boundary points produced using the Monte-Carlo-Simulation. At the end, predefined templates of windows and doors are used to extract the final boundaries.

Another application, in which the semantic information could be involved is the 3D reconstruction workflow. Indeed, 3D reconstruction contains the Structure from Motion and the Multi View Stereo tasks. Stathopoulou & Remondino (2019) introduced an approach in which they exploited semantically segmented images to support the typical SfM workflow and especially the time-consuming step, of the sparse feature matching. In addition, areas which are not useful for the typical SfM-MVS pipeline, such as the sky, were excluded from it, while they presented a transfer of image's labels from the 2D into 3D space i.e., 3D point cloud. More precisely, they exploited the powerful 2D image semantic segmentation methods, to improve the image matching procedure and to enrich the produced 3D points with a semantic information. Firstly, a Convolutional Neural Network (CNN) was trained, using a dataset, which was contain several images of historical buildings' facades. The training session ended with the remarkable results of 81% accuracy, 87% precision, 81% recall and F1 score 83%. Using the produced deep learning network images, which present facades, can easily be segmented into regions with semantic information. Thus, the semantically segmented images are used to some photogrammetric procedures such as matching. In fact, when a detected point, in the master image, searches to find its corresponding point in a candidate image, labelling constraints are used and so the searching procedure is limited to a specific region i.e., to the region which has same characteristics as the region in which the under-process point lies on the master image. Stathopoulou & Remondino (2020) introduced a method in which they exploit semantic constraints to improve the *merging step of the computed depth maps*, in patch based MVS approaches. More precisely, they aimed to implement a separate 3D reconstruction for each class while to improve the 3D output using semantic constraints and to remove erroneous matches, between points from different classes. Firstly, a sparse point cloud is produced using the known SfM workflow while each point is enriched with a semantic information. The sparse point cloud with camera positions and orientations, constitutes the input for the MVS pipeline. The semantic constraint is performed in the last step of the patch-based MVS algorithm. Thus, the extra semantic check is implemented during the MVS's *depth map fusion* step. Hence, this semantic check, exploits the transferred semantic information and eliminates matches between points from different classes, while a separate 3D reconstruction for each class, is executed using the semantic information. Using the *semantic-based MVS* algorithm, undesired classes such as, trees and reflective surfaces, can be excluded, and so an improved 3D model with reduced noisy parts, is produced.

References:

Adam, A., Chatzilari, E., Nikolopoulos, S. and Kompatsiaris, I., (2018). 'H-RANSAC: A hybrid point cloud segmentation combining 2D and 3D data' *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, *4*(2), pp.1-8.

Arapellis, O. 2020. Semiautomated edge detection on digital images. Report for the Postgraduate Course in Geoinformatics, NTUA (in Greek).

Ballard, D.H., (1981). 'Generalizing the Hough transform to detect arbitrary shapes.' *Pattern recognition*, *13*(2), pp.111-122.

Becerikli, Y., and Karan, T. M. (2005, June). A new fuzzy approach for edge detection. In *International Work-Conference on Artificial Neural Networks* (pp. 943-951). Springer, Berlin, Heidelberg.

Blomley, R., Weinmann, M., Leitloff, J., and Jutzi, B. (2014). 'Shape distribution features for point cloud analysis-a geometric histogram approach on multiple scales.' *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *2*(3), 9.

Canaz Sevgen, S., & Karsli, F. (2020). 'An improved RANSAC algorithm for extracting roof planes from airborne LiDAR data'. *The Photogrammetric Record*, *35*(169), 40-57.

Canny, J. F. (1983). *Finding Edges and Lines in Images* (No. AI-TR-720). *MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB.*

Canny, J. F. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679-698.

Davis, L. S. (1975). 'A survey of edge detection techniques.' *Computer graphics and image processing*, *4*(3), 248-270.

Dolapsaki M. (2020) 'Development of an algorithm for the detection of edges in point clouds using digital images' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (in Greek).

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). 'A density-based algorithm for discovering clusters in large spatial databases with noise'. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, pp. 226–231

Fischler, M.A. and Bolles, R.C., (1981). 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography' *Communications of the ACM*, *24*(6), pp.381-395.

Grilli, E., Dininno, D., Petrucci, G., and Remondino, F. (2018). From 2D to 3D supervised segmentation and classification for cultural heritage applications. In *ISPRS TC II Mid-term Symposium "Towards Photogrammetry 2020"* (Vol. 42, No. 42, pp. 399-406).

Grilli, E., Farella, E. M., Torresani, A., and Remondino, F. (2019). 'Geometric features analysis for the classification of cultural heritage point clouds.' *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Grilli E. (2019). 'Automatic classification of architectural and archaeological 3D Data' Ph. D Dissertation, University of Bologna, Bologna Italy.

Hough, P.V., (1962). '*Method and means for recognizing complex patterns.'* U.S. Patent 3,069,654.

Kanade, T. (1980). 'Region segmentation: signal vs semantics' *Computer Graphics and Image Processing*, *13*(4), 279-297.

Kirsch, R. A. (1971). 'Computer determination of the constituent structure of biological images' *Computers and biomedical research*, *4*(3), 315-328.

Kitchen, L., & Rosenfeld, A. (1982). 'Non-maximum suppression of gradient magnitudes makes them easier to threshold.' *Pattern Recognition Letters*, *1*(2), 93-94.

Kohler, R. (1981). 'A segmentation system based on thresholding'. *Computer Graphics and Image Processing*, *15*(4), 319-338.

Kroon, D., (2009). 'Numerical optimization of kernel based image derivatives' *Short Paper University Twente*.

Kroeger *P*. (2019). 'Analyzing Meaning'. *Language Science Press. pp. 4–6. ISBN 978-3-96110-136-8.*

L. G. Roberts, (1965). 'Machine perception of three dimensional solids, in Optical and Electro-Optical Information Processing' *(J. Tippett, D. Berkowitz, L. Clapp, C. Koester, A. Vanderburgh, Eds.), M.I.T. Press, pp. 159-197.*

Lewis, D., (1976). 'General semantics' In *Montague grammar* (pp. 1-50). Academic Press.

Liu, X., Deng, Z., & Yang, Y. (2019). 'Recent progress in semantic image segmentation'. *Artificial Intelligence Review*, *52*(2), 1089-1106.

MacQueen J., (1967). 'Some methods for classification and analysis of multivariate observations' *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1, pp. 281-297.*

Maini, R., & Aggarwal, H. (2009). 'Study and comparison of various image edge detection techniques.' *International journal of image processing (IJIP)*, *3*(1), 1-11.

Marr, D., and Hildreth, E. (1980). Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167), 187-217.

Matsuyama, T., (1989). 'Expert systems for image processing: knowledge-based composition of image analysis processes.' *Computer Vision, Graphics, and Image Processing*, *48*(1), pp.22-49.

Mitropoulou K. (2017) 'Development of a procedure for the detection of planes and edges in unorganized point clouds' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (https://dspace.lib.ntua.gr/xmlui/handle/123456789/44820) (in Greek)

Mitropoulou, A., Georgopoulos, A., 2019. An Automated Process to Detect Edges in Unorganized Point Clouds, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-2/W6, 99-105, https://doi.org/10.5194/isprs-annals-IV-2-W6-99-2019, 2019

Murtiyoso, A., and Grussenmeyer, P. (2019). 'AUTOMATIC HERITAGE BUILDING POINT CLOUD SEGMENTATION AND CLASSIFICATION USING GEOMETRICAL RULES'. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Nguatem, W., Drauschke, M. and Mayer, H., (2014, August). 'Localization of Windows and Doors' *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *2*(3), p.87.

Nielson, H. R., & Nielson, F. (1992). '*Semantics with applications'* (Vol. 104). Chichester: Wiley.

Prewitt, J. M. (1970). 'Object enhancement and extraction.' *Picture processing and Psychopictorics*, *10*(1), 15-19.

Rosenfeld, A. and A. C. Kak (1976). 'Digital Picture Processing.' Academic Press, New York, p. 275 .

Rosenfeld, A., and Thurston, M. (1971). 'Edge and curve detection for visual scene analysis' *IEEE Transactions on computers*, *100*(5), 562-569.

Rosten, E. and Drummond, T., 2006, May. Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430-443). Springer, Berlin, Heidelberg.

Sander, J., Ester, M., Kriegel, H.P. and Xu, X., (1998). 'Density-based clustering in spatial databases: The algorithm gdbscan and its applications' *Data mining and knowledge discovery*, *2*(2), pp.169-194.

Shen, X., Darmon, F., Efros, A. A., & Aubry, M. (2020). RANSAC-Flow: generic two-stage image alignment. *arXiv preprint arXiv:2004.01526*.

Skentzos O. (2020) 'Development of an automated algorithm for detecting edges in point clouds' Diploma Thesis, School of Rural and Surveying Engineering, National Technical University of Athens, Athens (https://dspace.lib.ntua.gr/xmlui/handle/123456789/51122), (in Greek)

Skimage documents, (2021) Perwitt's transform [online]. Available at: https://scikit-image.org/docs/dev/api/skimage.filters.html (Accessed: 01 January 2021)

Sobel, I., and Feldman, G. (1968) 'A 3x3 Isotropic Gradient Operator for Image Processing', *presented at the Stanford Artificial Intelligence Project (SAIL) in 1968*

Speaks J. (2019). "Theories of Meaning", *The Stanford Encyclopedia of Philosophy (Winter 2019 Edition)*, Edward N. Zalta (ed.), *Metaphysics Research Lab, Stanford University*, Available at: https://plato.stanford.edu/archives/win2019/entries/meaning/ (Accessed: 21 December 2020)

Stathopoulou, E. K., and Remondino, F. (2019). 'Semantic photogrammetry: boosting image-based 3D reconstruction with semantic labeling.' *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, *42*(2), W9.

Stathopoulou, E. K., & Remondino, F. (2020). 'Multi view stereo with semantic priors.' *arXiv preprint arXiv:2007.02295*.

Wikipédia, (2021) Laplacian of Gaussian [online]. Available at https://en.wikipedia.org/wiki/Blob_detection#cite_note-Lin13JMIV-1 (Accessed: 02 January 2021)

Wikipédia, (2021) Robinson Compass [online]. Available at: https://en.wikipedia.org/wiki/Robinson_compass_mask (Accessed: 02 January 2021)

Wikipédia, (2020) Semantics [online]. Available at: https://en.wikipedia.org/wiki/Semantics (Accessed: 21 December 2020)

Yuheng, S., & Hao, Y. (2017). 'Image segmentation algorithms overview' *arXiv preprint arXiv:1707.02051*.

Ziou, D., & Tabbone, S. 1998. Edge detection techniques-an overview. Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii, 8, 537-559.

Zucker, S.W., (1976). 'Region growing: Childhood and adolescence' *Computer graphics and image processing*, *5*(3), pp.382-399.

3. Practical Implementation

For the present Diploma Thesis two datasets were used, the Temple of Demeter in Naxos and the Old Police Station in Rhodes. They were acquired for the fieldwork of previous projects and are described in the following.

3.1 Temple of Demeter, Naxos Dataset

The temple of Demeter at Gyroulas in Naxos, dates from 525 BC [Giannakoulas 2018] i.e., it was built in the Archaic Period, which lasted from the 7th to 6th c. BC i.e., between the Geometric and the Classic Period [National Archeological Museum, Archaic Period]. In 1949, N. Kontoleon found the ruins of an ancient temple, near the church of Aghios Ioannis in Sagri, Naxos which was constructed using ancient materials [Stefanou 2018]. The temple was revealed, during the archaeological excavation, which was conducted in 1954 AD, by N. Kontoleon. However, after his death, in 1976 [Stefanou 2018], the excavation was continued by V. Labrinoudakis in cooperation with the Polytechnic School of Munich and the architects, M. Korres, G. Gruben and A. Ohnesorg, and lasted until 1995 AD [Giannakoulas 2018]. The temple of Demeter is located approximately 11 kilometers SE from the village of Naxos i.e., roughly in the middle of the island [Figure 1].

Figure 1: Naxos island and the temple of Demeter (© Google Earth)

The Temple of Demeter as is partly reconstructed today was geometrically documented for two Postgraduate theses [Stefanou 2018 and Giannakoulas 2018] using contemporary digital techniques. In this diploma thesis, a limited number of images from the temple of Demeter at Gyroulas in Naxos, are used. The images are firstly enriched with edge semantic information and then are used into the SfM-MVS workflow using Agisoft Metashape or OpenSfM software. Some images of the temple of Demeter dataset, which are used in this diploma thesis, are depicted in Image 1. The images were captured using the, full frame DSLR, Canon EOS-1Ds Mark III camera, while their dimensions are 5616x3744 pixels.

Image 1: Four sample images of the Demeter temple dataset

3.2 Old Police Station, Monolithos, Rhodes Dataset

Monolithos is a village on the island of Rhodes with no more than 150 inhabitants. It is located approximately 72 km SW from the Rhodes city. The inhabitants of Monolithos village are mainly engaged in agriculture, livestock and beekeeping. A plethora of cultural heritage sites such as the castle of Monolithos and the ancient Kymissala, are situated in the area around Monolithos. Information about the ancient Kymissala can be found in Stefanakis et. al. (2015). Thus, Monolithos, as well as the other nearby villages, arouses the interest of scientists, from several scientific fields, such as archeologists and surveyors. The old police station in Monolithos, was constructed by the Italians, and was used as the local police department. Nowadays, the old police station, is used by the archeologists of the University of the Aegean, who take part in the archeological investigations and excavations in ancient Kymissala. In Figure 2 the island of Rhodes, the Rhodes city, the Monolithos village as well as some other villages are depicted. In Figure 3 the village of Monolithos and a top view (red square) of the old police station are depicted.

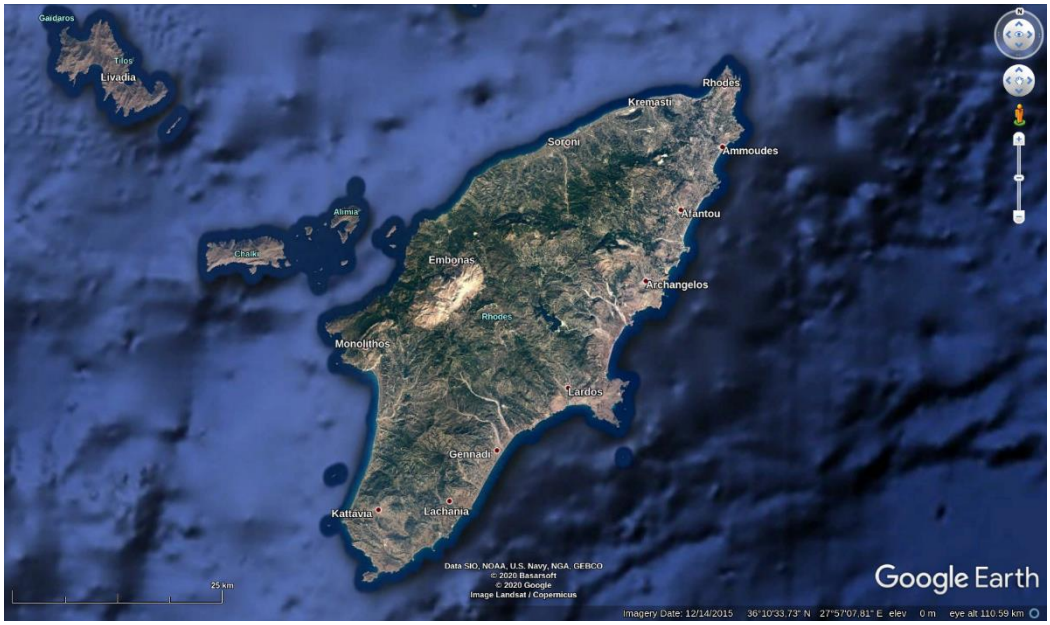


Figure 2: Rhode Island and Monolithos village (© Google Earth)

Figure 3: Monolithos Village and Old Police Station (© Google Earth)

In this diploma thesis, a limited number of images from the Old Police Station, in Monolithos are used. The images were captured during the summer field course in photogrammetry in 2018, which was conducted by the Laboratory of Photogrammetry SRSE NTUA. The images are firstly enriched with edge semantic information and then are used into the SfM-MVS workflow using Agisoft Metashape or OpenSfM software. The images were captured using a Canon EOS 80D camera, while its dimensions are 6000x4000 pixels. The front view of the Old Police Station is depicted in Image 2.

Image 2: Old Police Station front view (© Google Earth)

3.3 Structure from Motion - Multi View Stereo and other Software

3.3.1 Agisoft Metashape:

Agisoft Metashape is a commercial software which was introduced in 2006 as Agisoft PhotoScan (Agisoft Metashape 2021). It is one of the leading photogrammetric software on the market and thus it is used in a plethora of applications and researches. For instance, it is used for photogrammetric triangulation, dense cloud production, 3D model production, orthophoto creation, panorama stitching etc. In addition, it supports a Python and a Java API and so it can be combined with several add-ons as well as it can manage the provided algorithms-workflows using those programming languages. In this diploma thesis the Agisoft Metashape photogrammetric pipeline is used up to the dense cloud production step. Briefly, the images are inserted and aligned by executing an SfM algorithm and the sparse point cloud in combination with the image poses are extracted. Afterwards an MVS algorithm is performed and the dense point cloud is produced. In fact, the complete pipeline, i.e., up to the production of texture models and the creation of orthophotos, is a perplexing and demanding task while in most of the cases a post processing procedure of the produced mesh i.e., triangles, is essential to fill up the mesh's holes and to produce an accurate and realistic 3D model. However, the complete pipeline is out of the scope of this diploma thesis and thus it is not further analyzed. Finally, a crucial advantage of Agisoft Metashape software is its ability to process multispectral images, such as from satellite sensors, and thus, in combination with its professional performance, it constitutes a valuable choice.

3.3.2 Mapillary - OpenSfM

The OpenSfM (OpenSfM 2021) is an open-source structure-from-motion pipeline, sponsored by Mapillary which is an organization with a global network of members, in 190 countries. Hence, Mapillary, exploits the data, which are collected from its members, to produce better maps and in general to visualize the world. Mapillary, has its data open as well as its software, such as OpenSfM. OpenSfM is distributed under the BSD-2-Clause license. The source-code as well as the license can be found in GitHub (OpenSfM source-code 2021). OpenSfM is written in Python while it is accessible using the computer's terminal. More concretely, after the software's building, the user can execute the SfM-MVS workflow for each case study, creating a new directory into the folder data. Then, using a sequence of commands one can produce firstly a sparse and then a dense, point cloud. Finally, the results can be visualized into a JavaScript viewer, which is included by default. Apart from the basic SfM pipeline, the OpenSfM software provides several features to improve the quality of its results, such as the exploitation of GPS or accelerometer measurements to manage the matching procedure in a faster way. OpenSfM has

the advantage of an open-source software, which is the ability to adapt the source-code to one's needs, as well as to understand its operation in depth, i.e., it is not a "black box". On the other hand, OpenSfM is not supporting multichannel images. In fact, most of the applications are conducted using common images i.e., RGB but for this diploma thesis, the SfM-MVS pipeline must support multichannel images and thus it constitutes a difficulty. Additionally, the user must define a lot of SfM-MVS parameters, to achieve high performance results.

3.3.3 AliceVision – Meshroom

Meshroom is an open-source 3D reconstruction software supported by ALICEVISION association, a non-profit organization, founded in July 2020, which "aims to *democratize 3D digitization technologies from photographs"* (Meshroom 2021). Both academic and industry institutions, have been involved in the development of Meshroom while it is distributed under the MPL2 license. It comes with a friendly UI while providing a powerful graph editor in which the user can tackle the 3D reconstruction task using several approaches. The UI consists of seven windows, the (i) Images, (ii) Image Viewer, (iii) 3D Viewer, (iv) Graph Editor, (v) Node, (vi) Display, and (viii) Scene. It is worth noting that the image viewer window could display the extracted features as well as several statistics during the process. The default graph editor structure contains eleven nodes which are (i) CameraInit, (ii) FeatureExtraction, (iii) ImageMatching, (iv) FeatureMatching, (v) StructureFromMotion, (vi) PrepareDenseScene, (vii) DepthMap, (viii) DepthMapFilter, (ix) Meshing, (x) MeshFiltering and (xi) Texturing. A lot of configurations are provided for each node. For instance, Meshroom provides several feature extraction methods such as SIFT and A-KAZE or a lot of selections for the SfM algorithm configuration like the ability to define the maximum reprojection error, which will be accepted during the reconstruction procedure. The previously described features, as well as additional ones not mentioned in this effort, constitute Meshroom a professional 3D reconstruction tool. The advantages of Meshroom are several, like that is not a "black box", due to its open-source license or the provided informative and user-friendly UI etc. On the other hand, multichannel images processing is not supported from the Meshroom software which constitute a software's disadvantage for the purposes of this diploma thesis. Additionally, it requires Graphic Cards with CUDA, thus limiting them to NVDIA brand.

3.3.4 CloudCompare

CloudCompare is a powerful open-source software which is used for point cloud processing tasks. It has begun as a Ph.D. thesis (D. Girardeau-Montaut (2006)) on change detection on 3D geometric data, in 2006. It is written in C++ while it provides various algorithms and features for point cloud manipulation tasks. For instance, CloudCompare provides, point cloud merging, mesh editing and point cloud segmentation techniques. Additionally, it calculates point cloud statistics, normal vectors etc. In this diploma thesis, the CloudCompare software is used as a 3D viewer to visualize the produced point clouds, to evaluate the detected edges as well as to display the segmented point cloud when it is necessary to. Finally, using the CloudCompare software it can manipulate labelled data i.e., 3D points, and to visualize them with different approaches. Most of the 3D visualizations in this section, were conducted using the CloudCompare software.

3.4 Developed Approaches

The aim of this diploma thesis is to extract 3D edges, from a point cloud, using edge semantic information. The point cloud is produced using image-based approaches i.e., SfM-MVS workflows. The edge semantic information on the images is created using an appropriate edge detection technique. Then, each RGB image is enriched with a label channel containing its edge semantic information i.e., RGBL. Finally, these images are introduced into the SfM-MVS workflow. The output is a point cloud in which

each point is associated with a label value. Thus, the points are classified into edge points and all the rest, using their label values. Then, the separated edge points, are further classified into points of each line. Finally, to each set of points, a line is fitted while a vector (e.g., ".dxf") archive is produced, containing the vectorized 3D lines. The general idea of the proposed approach is visualized in Image 3.

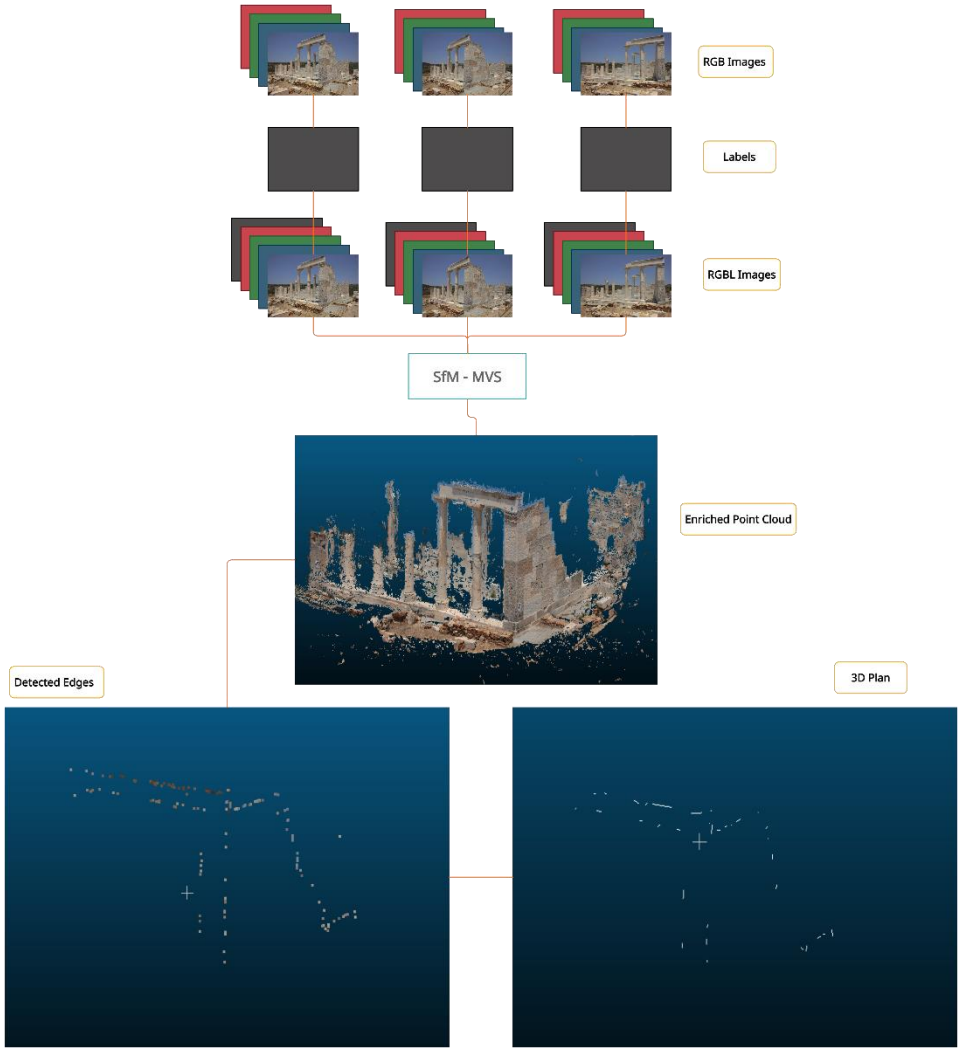
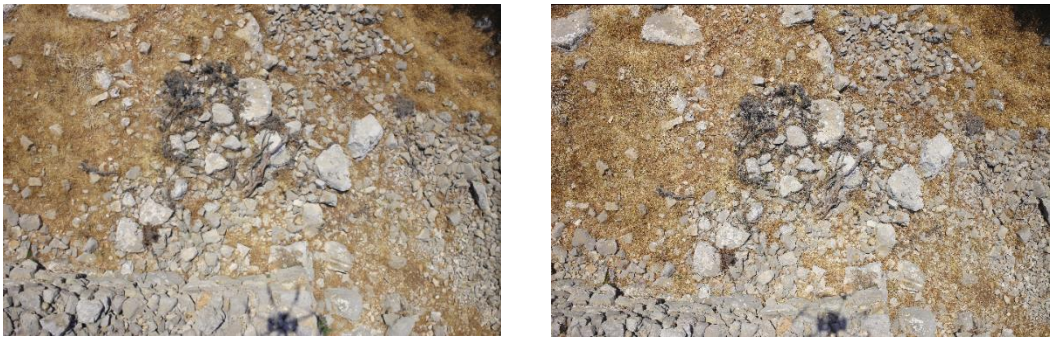


Image 3: Proposed Approach

3.4.1 Implementation using the MyTriangulation pipeline

Firstly, a python triangulation script, called "MyTriangulation", was developed to investigate the principal idea of the proposed method i.e., to enrich an image with a label channel and then pass the semantic information into 3D space. "MyTriangulation" simplifies the complicated SfM-MVS approach into the standard two images epipolar geometry problem, in which a non-optimized 3D point cloud is generated. "MyTriangulation" script consists of one class, called "Triang", which aims to manipulate the entire procedure. The major steps of the "MyTriangulation" pipeline are, (i) Reads the images, (ii) Extracts images' features, (iii) Finds the image pairs (iv) Matches each image pair, using a Flann based matcher (Muja & Lowe 2009) as well as the Lowe's ratio (Lowe 2004), (v) Calculates the essential matrix, if the intrinsic parameters are known, otherwise calculates the fundamental matrix, (vi) Finds the right image's rotation and translation matrix with respect to the master image i.e., left, (vii) Finds the projection matrix, (viii) Triangulates the images and exports the generated point cloud. The change between the fundamental and the essential matrix is performed manually i.e., comment-uncomment.

Also, the pairs are created by associating each image with all the other ones e.g., for 3 images the pairs are (0, 1), (0, 2), (1, 2) and thus it is computationally inefficient. However, the presented disadvantages were not significant to affect the results of the first investigation because it was not the goal of this approach to achieve the lofty standards of a commercial SfM-MVS workflow. Additionally, only one pair of images could be used. Image 4 depicts the two images which were used for the "MyTriangulation" implementation. These two aerial large scale images are from the ancient Kymissala, in Rhodes while they were captured during the summer field course in photogrammetry in 2019, which was conducted by the Laboratory of Photogrammetry SRSE NTUA.



Left Image Right Image

Image 4: Image Pair for "MyTriangulation" experiment

To handle the first two steps a python script called "Geometry" was produced, in which the class called "Image" was defined. This class calculates the image's camera matrix and extracts the image's feature points. Firstly, the image's channels are extracted and stored. Then the image's camera matrix is calculated. If the intrinsic parameters are known, they are extracted from the image's metadata and then the camera matrix is determined, otherwise an approximation of the camera matrix is created. In fact, the camera matrix is a 3x3 matrix in which the intrinsic parameters are involved. In more detail, the camera matrix represents the shape of the bundle of rays i.e., the camera's geometry. The parameters, involved in the camera matrix calculation, are the x and y component of the focal length (f_x, f_y) and the position of the principal point (p_x, p_y) as well as the skew parameter a. The generalized version of the camera matrix is presented in Image 5. For this implementation, the skew parameter a, equals to 0 while the f_x, f_y, $p_x$and $p_y$are extracted from the image's EXIF.

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{f_x} & \boldsymbol{a} & \boldsymbol{p_x} \\ \boldsymbol{0} & \boldsymbol{f_y} & \boldsymbol{p_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Image 5: Camera Matrix

Afterwards, the image's feature points are extracted. The implemented feature extraction methods are based on the A-KAZE (Alcantarilla & Solutions 2011), SIFT (Lowe 2004), SURF (Bay et. al. 2006) or ORB (Rublee et. al. 2011) algorithms. A point cloud produced using SIFT algorithm and one using A-KAZE, are depicted in Image 6 and 16, respectively. In Table 1 a comparison between the implementations of those algorithms, is presented.

Table 1: Comparison between the implemented edge extraction techniques

Algorithm	Execution Time (sec)	Number of Points
A-KAZE	6	220.000
SIFT	26	500.000
SURF	6	45.000
ORB	1	4.000

The steps 1 and 2 are executed for each of the given images i.e., image 1 and image 2, according to this implementation. Thus, a new “geometry” or structure, for each image, is produced in which each of them is associated with its camera matrix and extracted feature points. The rest of the steps are performed using the “Triang” class i.e., the “MyTriangulation” script.

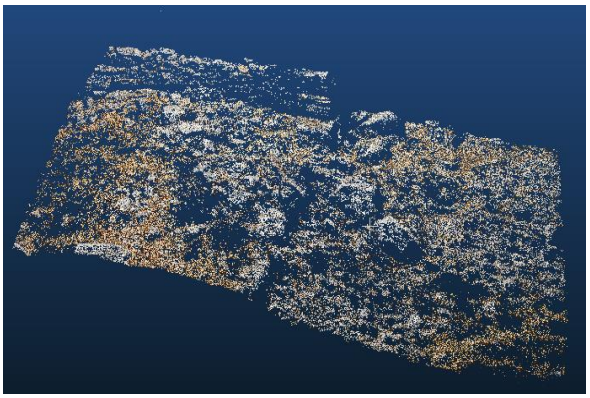


Image 6: MyTriangulation point cloud using SIFT

Since the developed workflow was able to produce a 3D point cloud, the first try using semantic information was implemented. The process was begun using the RGB images and one image containing the semantic information. To further simplify the first idea, which was to enrich the images with the semantic information using an additional channel, the left image was duplicated, and the semantic information was manually added on the duplicate, using a drawing software. To be more specific, three stones were painted yellow, light purple and red respectively, as well as some lines were drawn, using dark purple color. The first approach was implemented by applying the “MyTriangulation” workflow using the two images but instead of coloring each point using the given images, the colors were extracted from the segmented image i.e., the drawn copy of the left one. In fact, the points colors, exploiting the professional SfM-MVS approaches, are calculated either using the left image’s corresponding color or calculating the mean value between the images’ color values, from which each point was determined. Hence, in this approach it is just taken from the copied image. Image 7 depicts the image, which contains the semantic information.

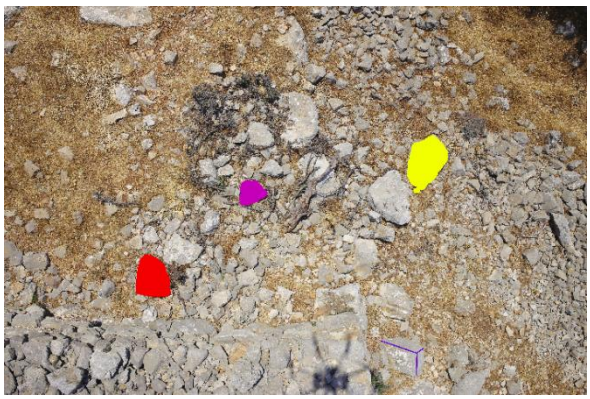


Image 7: Image with manually added semantic information

The produced point cloud is depicted in Image 8, in which the corresponding, to the semantic information, points are distinguished by their color.

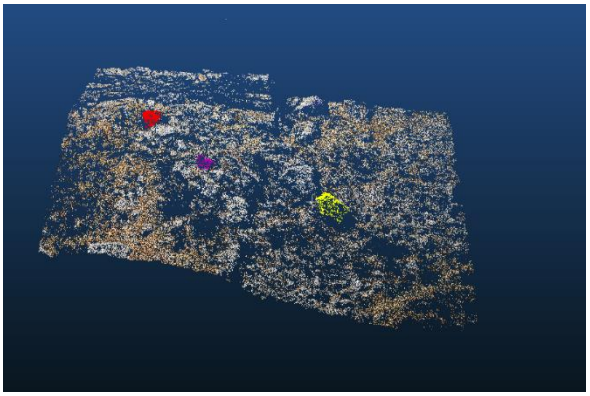
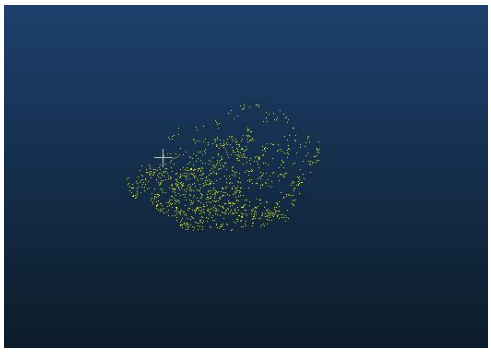
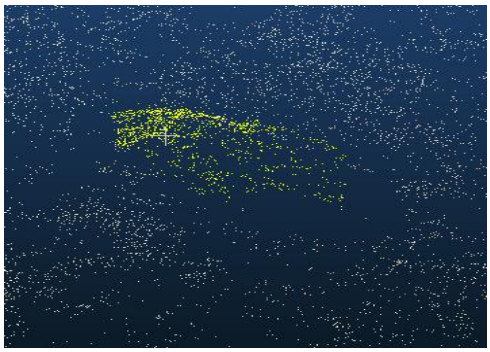


Image 8: Point cloud with colored elements

In a post process analysis, each class' points were separated, using their color. For instance, the yellow rock was extracted, detecting the yellow points in the produced point cloud. In Image 9 the detected rock is depicted as well as the rock inside the 3D point cloud (different viewpoint).



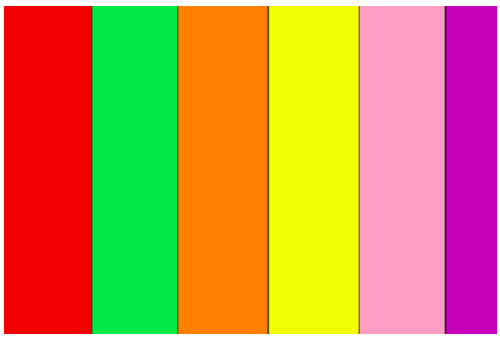
Yellow Rock



Yellow Rock with Point Cloud

Image 9: 3D points detection using color value

Finally, the color manipulation, during the "MyTriagulation" pipeline as well as the point cloud itself, were evaluated. The color manipulation was checked using an external image, which had the same shape as the original ones, but it had a solid color structure i.e., the colors were easily distinguished (Image 10). The same procedure as the previously described one i.e., rocks and lines extraction, was implemented. Point cloud colors play a significant role to construct an improvement for the "MyTriangulation" pipeline, in which the images will be enriched with a semantic channel, because the labels are manipulated with the same approach as the colors do. The generated point cloud contains the correct colors, as the Image 10 depicts.



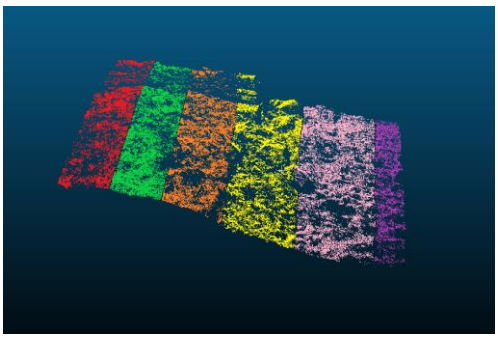
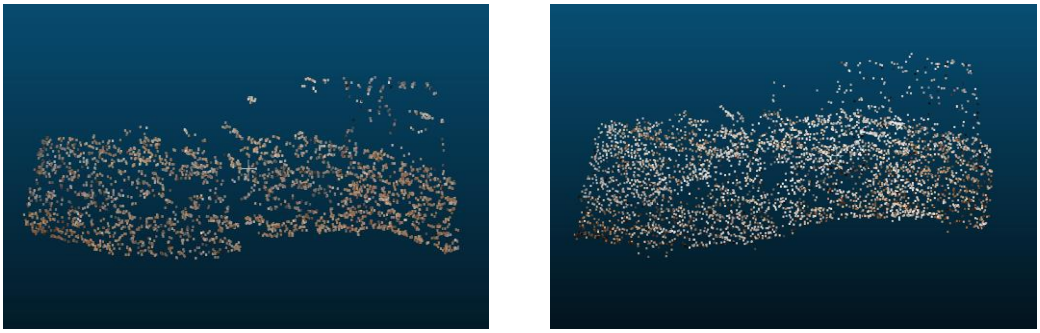


Image 10: MyTriangulation point cloud color check

To check the point cloud itself, two sparse point clouds were generated using Metashape and Meshroom software (Image 11). This check revealed that the generated point cloud was visualized in mirror view in comparison with the point clouds which were generated using professional software and thus the pipeline was improved to handle this drawback. The corrected point cloud is presented in Image 12 while it was produced using A-KAZE algorithm.



Metashape Meshroom

Image 11: Point Clouds using professional software

Image 12: Corrected “MyTriangulation” point cloud using A-KAZE

The crucial disadvantages of this approach are the production of a non-refined point cloud i.e., with no bundle adjustment step included as more than two images are needed, as well as the constraint to use the pipeline only with a pair of images at a time. Apart from them, one more drawback occurs, which is the replacement of the semantic objects’ real colors, with the annotated ones, as the previous example presents. Additionally, it is inefficient to duplicate images as well as to overload the entire process with them. To overcome the color disadvantage a second approach was implemented, in which the images are enriched with the semantic information using an additional label channel. Hence, each image is transformed into a 4-channel one, carrying the red, green, blue, and the label channel, i.e., RGBL. To achieve that, a script called “SemanticPass” was generated, to produce the images which will be inserted into the “MyTriangulation” pipeline. The “SemanticPass” script consists of one class named “SFMImage”. During this implementation two images from the temple of Demeter dataset were selected (Figure 4). Then, manually annotated semantically enriched images were produced (Figure 4) while the masks in Figure 4, were finally used as images’ semantic information. The masks were produced mapping the red pixels in the manually annotated images with mask’s corresponding pixels, and then defining them as white. Firstly, “SFMImage” class reads the under-process image while it calculates its characteristics e.g., width, height, etc. Then, the image is separated into its principal channels i.e., red (R), green (G), and blue (B).

Figure 4: Final semantic information

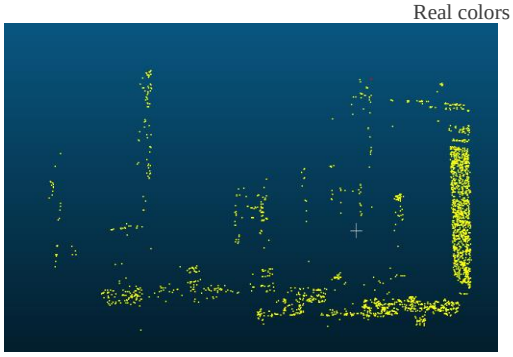
Afterwards, “SFMImage” class reads and stores the semantic channel, which is associated with the under-process image. In fact, there are two methods to read and store the semantic information. If the given label image has one channel, like the masks (Figure 4), the “SFMImage” class simply reads and stores the semantic channel. If the given image has three channels, like the manually annotated images in Figure 4, the label channel is produced separating the annotated objects i.e., it produces the masks by itself. After that, the under-process image is enriched with the label channel using the “add_channel” function, in which the R, G, B and L channels are merged into a 4D array. Finally, the “SFMImage” class saves the enriched images using for each one the three-channel image’s name and the .tiff format (pseudo-tiff images are produced, if the given ones are not in “.tiff” format). To evaluate the "add_channel” function a check pipeline i.e., reverse process, was performed. Firstly, a three-channel image was enriched with the label channel and saved as .tiff image. Then the four-channel image was read again, and its channels were separated. The R, G and B channels were merged and then saved while the label channel was saved separately. Afterwards, the produced RGB image was compared with the original one and the produced label channel, was compared with the given label channel. The check was carried out successfully and thus the semantic implementation continued. The produced images, which were enriched with the edge semantic information, are fed into the “MyTriangulation” workflow and the labeled point cloud was generated (Image14). The produced point cloud contains seven registered values, for each generated point, which are the X, Y, Z, R, G, B and L (Image 13). If the L equals to 255 (8-bit

image), the point corresponds to an annotated object i.e., the drawn lines. Thus, the points were separated into those which have label value 255 and all the rest. The produced point cloud, with its real colors as well as with the label colors, is depicted in Image 14.

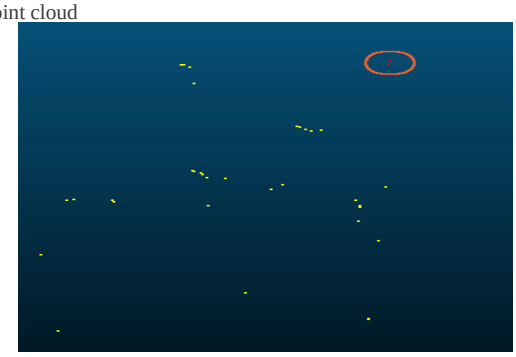
```
-1.1065379568417553 2.331600030717536 -0.5886231143259473 112 97 92 0
-0.9036277349956882 2.38844001889179 -0.6043160077271645 124 104 95 0
0.18180238867549067 4.274673517855274 -1.1217154088503836 204 196 194 255
-1.0357650983773445 2.35178915802395 -0.5797790350288865 138 115 107 0
```

Image 13: Semantic point cloud's registered values

Real colors point cloud



Label colors point cloud



Zoom in labeled points (inside orange ellipse)

Image 14: Enriched point cloud analysis

Unfortunately, the points which had a label value 255 were only two, due to the weakness of the "MyTriangulation" workflow. The weakness of the "MyTriagulation" process, is also indicated by the non-detection of the front annotated lines despite the dense production of the points in their surrounding area. In conclusion, a professional SfM-MVS workflow, should be used either to address the previously described weaknesses, or to produce a highly reliable point cloud and finally extract the 3D lines with the most feasible accuracy.

3.4.2 Implementation using the OpenSfM software

OpenSfM is an open-source SfM software, which contains python bindings, while it operates using terminal commands. The software's repository was downloaded from GitHub. Then it was built and a dense point cloud, using the included dataset called "berlin", was produced. OpenSfM documentation was very helpful while the installation was conducted without any issues. It is worth noting that the OpenSfM was installed, on an Ubuntu 20.04.1 LTS (Focal Fossa) operating system. Afterwards, an implementation using 47 images from the temple of Demeter, in Naxos dataset, was executed applying three steps. Firstly, a new directory, called "TempleOfDemeter" and one called "images", were created inside OpenSfM's "data" directory and inside "TempleOfDemeter" directory, respectively. Secondly, the 47 images as well as the archive "config.yaml", from the "berlin" directory, was moved into the "images" and "TempleOfDemeter" directory respectively. OpenSfM uses a "config.py" script in which a plethora of useful variables, for the SfM-MVS procedure, are defined. In fact, the user interacts with

the pre-defined variables i.e., to activate or change them, using the "config.yaml" archive. Firstly, the SfM-MVS pipeline was implemented using the default parameters, which are displayed in Image 15 while the produced point cloud is depicted in Image 16.

```
processes: 8                  # Number of threads to use
depthmap_min_consistent_views: 2      # Min number of views that should reconstruct a point for it to be valid
depthmap_save_debug_files: no         # Save debug files with partial reconstruction results
feature_process_size: 1024
bundle_use_gcp: yes
```

Image 15: OpenSfM default parameters

Image 16: OpenSfM's point cloud using the default parameters

The produced point cloud is noisy e.g., a lot of sky points were generated along the temple's edges and in general unreal points occur. Thus, the default parameters were redefined (Image 17).

```
1
2 # OpenSfM will use the default parameters from opensfm/config.py
3 # Set here any parameter that you want to override for this dataset
4 # For example:
5 processes: 8                  # Number of threads to use
6 depthmap_min_consistent_views: 3      # Min number of views that should reconstruct a point for it
  to be valid
7 depthmap_save_debug_files: no         # Save debug files with partial reconstruction results
8 feature_process_size: 1024
9 feature_type: AKAZE
10 flann_algorithm: KDTREE
11 lowes_ratio: 0.7
```

Image 17: Parameters which was used with A-KAZE algorithm and OpenSfM

First and foremost, the "depth_min_consistent_views" was set to 3. This parameter, as the comment presents, defines "*the minimum views that should reconstruct a point for it*" and thus using three views a quite fine result could be revealed. Then, the feature extraction method was defined as "AKAZE". In addition to the default parameters, some extra were included such as the Lowe's ratio, which was redefined as 0.7, instead of 0.8, to improve the matching procedure. The OpenSfM pipeline was executed again and the produced point cloud is displayed in Image 18.

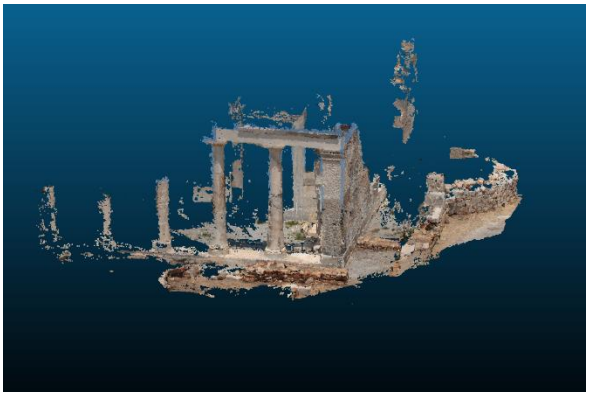


Image 18: OpenSfM's point cloud using AKAZE

Finally, one more implementation, using the SIFT algorithm in combination with a new set of parameters, was executed. The parameters are depicted in Image 19 while the determined point cloud in Image 20.

```
# OpenSfM will use the default parameters from opensfm/config.py
# Set here any parameter that you want to override for this dataset
# For example:
processes: 8                  # Number of threads to use
depthmap_min_consistent_views: 3      # Min number of views that should reconstruct a point for it
to be valid
depthmap_save_debug_files: no         # Save debug files with partial reconstruction results
feature_process_size: 1024
feature_type: SIFT
flann_algorithm: KDTREE
lowes_ratio: 0.7
```

Image 19: Parameters which was used with SIFT

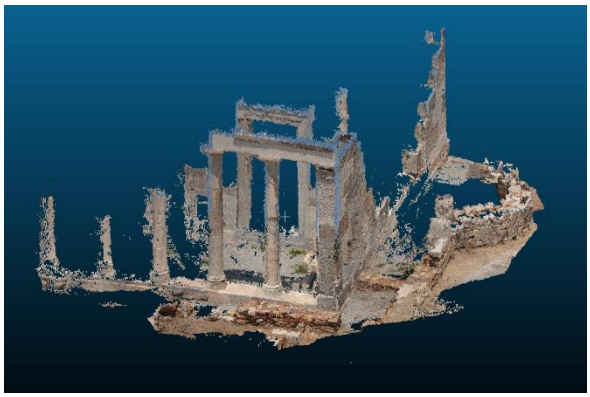


Image 20: OpenSfM's point cloud using SIFT

The produced point clouds using the AKAZE or SIFT algorithm, were quite fine, for the demands of this diploma thesis. Of course, a post process procedure of those point clouds e.g., noise removal, will be resulting finer outputs. In this effort a further process of the produced point clouds was not implemented. In this diploma thesis the SfM-MVS pipeline is fed with 4D images i.e., RGBL, thus, an implementation using such images as well as OpenSfM workflow, was executed. Before the MVS execution, OpenSfM workflow undistorts the given images as the theory dictates. The default pipeline saves the given images using the ".jpg" format, and hence loses the fourth channel. Thus, the undistorted images format was replaced using the ".tiff" one. Despite that the undistorted images inherited the fourth channel from the given images i.e., the semantic information, the produced point cloud was the same as using the RGB imageries. Due to that, the OpenSfM source code was modified to pass the label information through the SfM-MVS workflow and to enrich the produced point cloud with it.

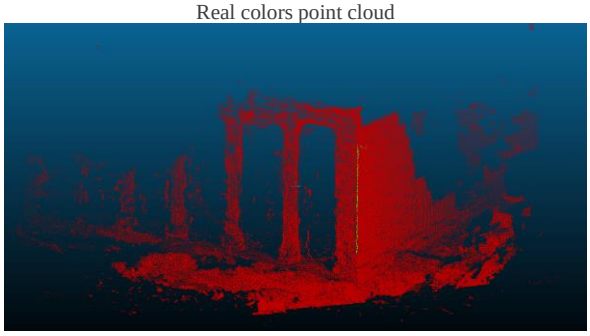
During the modification of the OpenSfM's source code the Git version-control-system was used. *Git is an open-source version-control-system, which stores and facilitates the use of various snapshots of user files over time* [Git]. *It thinks the files and archives as a sequence of snapshots* [Git]. Each snapshot is a picture of the file system in a specific moment. Using Git, while a software is produced, facilitate the procedure to store different versions of the same software, with an efficient way and the ability to retrieve them if necessary. To this end, the files as well as their versions, are protected and efficiently approachable. Additionally, Git provides the ability to multiple programmers to modify simultaneously a project, via separated branches. A branch is a separated instance of the same program which provides the ability to each programmer to manage it differently than its colleagues. Finally, all or some of the branches can be merged to the master one resulting the final product. Git is not limited to these abilities, instead is a powerful tool with many other benefits. Git's commands are executed directly using the terminal. Firstly, the user initializes the Git directory, which is invisible i.e., hidden, using the "git init" command into the under-process directory. When a script is modified the recent version of it, is stored using before the modification the "git add [name]" command, which enables to track the file during the process, and finally the "git commit –m [commit title]" command, which stores the modified tracked archive or archives and adds to the process a descriptive title i.e., the "commit title". Thus, for the modification of OpenSfM source code a new branch was created to store the modified scripts and its versions as well as to protect the master branch i.e., the original one. Additionally, at the end of the process the first and final, commits could be compared to each other to identify the final modified areas and so to concentrate on the changes and not on the documentation of them during the process.

The main idea of the modification process was to edit the OpenSfM workflow to identify if a 4 channel image occurs, then to process the labels i.e., the fourth channel, in the same way as it processes each color channel, and finally, to include each point's label value into the exported point cloud. Firstly, the function "extract_features", which is included into the "features.py" script, was edited to identify if an image contains four channels. Then it excludes the fourth channel during the feature extraction process. Instead, the semantic information is included into the extraction of feature's color values i.e., the produced color list contains four elements instead of three. Additionally, the format of the undistorted images was defined as ".tiff" i.e., the corresponding variable, into the "config.py", was simply changed from ".jpg" to ".tiff". The first 4D approach, which was described before, was resulted, that using the ".tiff" format, the image's fourth channel was inherited to the undistorted images, which are used into the MVS workflow, but the semantic information was not delivered to the produced point cloud. Thus, some functions were further modified. Firstly, the function "add_views_to_depth_pruner", which is included into the "dense.py" script, was edited. To be more specific, a variable called "labels" was defined, while it was associated with the 4^{th} channel of the under-process image. Then, the "labels" channel was scaled down in the same way as all the rest channels. In fact, OpenSfM contains a variable called labels by default, which is used in combination with an add-on script, for manually image annotation tasks, but in these implementations this variable is not used and so its position into the entire process, including the exported point cloud, is exploited when it is necessary. Hence, the point cloud's structure contains a position, called "class" for the classes, which are produced using the add-on, for each point. This position is used to attach in it, the edge semantic information i.e., the fourth element from the edited color list, for each point. Finally, a point cloud, enriched, with the edge semantic information, is produced called "merged.ply" in which eleven values are presented. The tenth value corresponds to the semantic information. It is worth noting that, the given semantic values are "255" (8-bit images) or "0" but during the process the OpenSfM workflow resizes the given images using an interpolation method. Thus, the values are changed, i.e., they vary between "0" and "255". The changed values constitute a crucial problem during the classification procedure, if the goal is to identify more than two classes, instead of just "edges" and "no-edges". More concretely, the detection of each class is transformed into a hard process due to the difficulty to identify each class along the mixed values, which are produced applying the interpolation method. In this diploma thesis, the changed values can be exploited during the classification problem because, the larger values indicate "stronger" lines instead of the smaller ones. Hence, a more accurate output, can arise, eliminating the weak edges during the 3D edge detection procedure.

After the described modifications, an implementation using five images from the temple of Demeter dataset, was performed. A line was drawn manually to each image. The RGBL images were constructed, via the “SemanticPass” script, and fed into the OpenSfM workflow. The RGB images as well as the manually annotated ones are depicted in Image 21. The produced enriched point cloud is depicted in Image 22, using two visualizations. In the first one the point cloud is visualized using the real colors while in the second one using red and green colors. The green points constitute the detected edge points i.e., the points which were generated from the annotated edge.

Image 21: RGB images and manually annotated ones

Real colors point cloud



Semantic colors point cloud

Image 22: Enriched point cloud visualizations

Afterwards, the points were further classified into edge’s points (Image 21) and all the rest, while the edge’s points were used for a vectorization task. The detected points were stored as “.ply” and “.txt” archive.

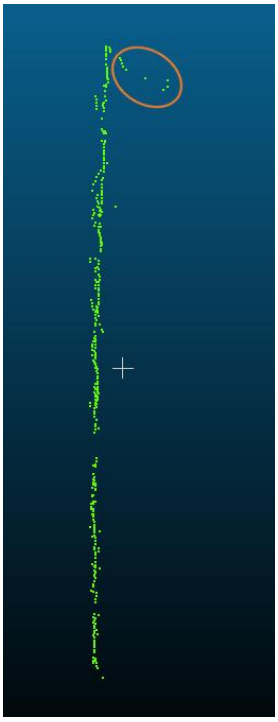
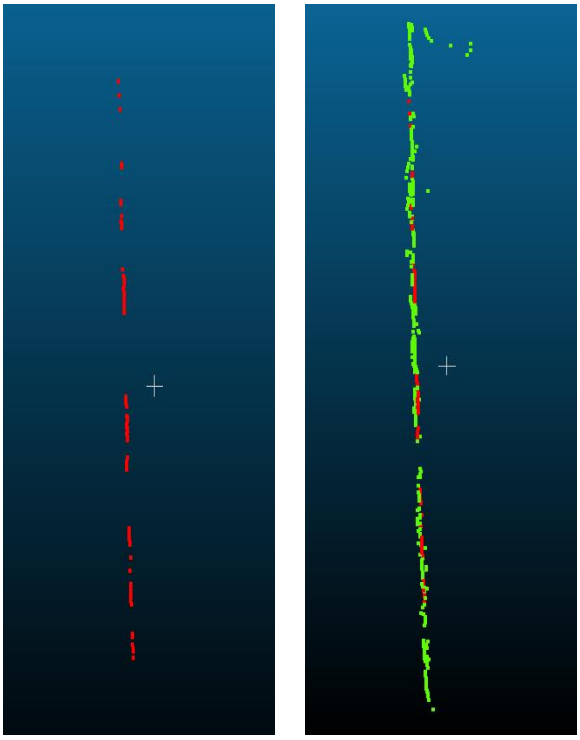


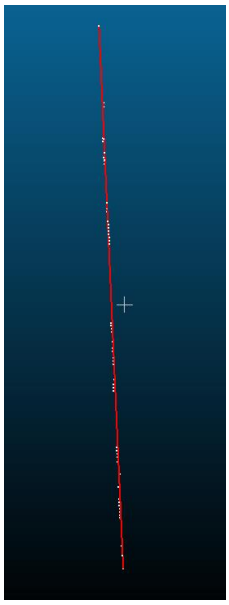
Image 23: The extracted edge points using OpenSfM

The points do not lie on the same line, as Image 23 depicts. Furthermore, some points (inside the ellipse) constitute miss-detections due to the width of the drawn line. However, the majority of the detected points construct a valid set from which a line can be determined. Thus, a further process which aims to vectorize the detected line, using the entire set of points, was conducted. Firstly, the RANSAC algorithm was executed to separate the inlier from the outlier points i.e., miss-detections. To execute the RANSAC algorithm three parameters must be defined (i) the minimum samples, (ii) the threshold value and the (iii) algorithm's iterations. The minimum samples parameter was defined as 2 because the process aims to detect a line and thus the minimum number of points which can define a line are two. The threshold value was set as 0.003 i.e., if the distance, between a candidate point and the candidate line is smaller than 0.003 units the point is characterized as inlier otherwise as outlier. The inliers are depicted in Image 24. The vectorized line in combination with the inliers as well as with the generated point cloud are depicted in Image 25. The vectorized line was stored using the ".dxf" format.



Inliers Extracted points and Inliers

Image 24: Inliers using RANSAC



The inliers and the generated line

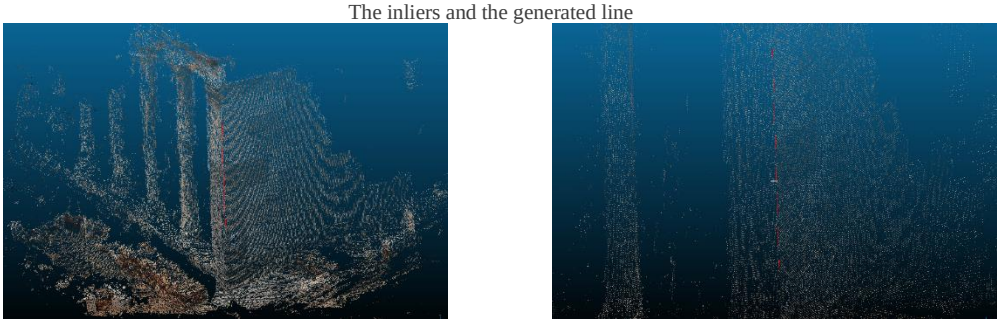


Image 25: 3D Line using OpenSfM

In fact, the user cannot manually annotate all the edges of a structure in every image. Thus, this approach is not efficient under real conditions. To cope with this issue an automate edge detector was used. To be more specific, the Canny algorithm was applied to detect the edges in each image. In general, Canny edge detector is highly dependent on each image's conditions i.e., for each image the Canny's parameters are different even if the images depict the same objects. In this implementation the parameters were the same for each image and thus additional noise was present into the produced label channels. The implementation of the Canny edge detector was included into the "SemanticPass" script. Hence, three conditions can be presented during the insertion of an RGB image in the "SemanticPass" script, (i) the given labels consist of one channel and thus it is directly merged with the given images, (ii) the given labels are manually annotated RGB images and thus 1D masks are produced and merged with the given images and (iii) only the RGB images are available i.e., there are not available the labels like the previous cases, and thus the Canny edge detector is applied. Into condition (iii) the RGB images are inserted one by one, into the "SemanticPass" script, the Canny algorithm is applied, and an edge map is generated for each of them. Then the RGB channels are merged with the produced mask i.e., label channel, and the RGBL images are saved. Finally, the enriched images are fed into OpenSfM's workflow. In general, the dense point clouds consist of millions of points from which the direct methods aim to detect 3D edges. The enriched point cloud reduces the number of the points into thousands i.e., excludes the redundant information. Thus, the point cloud which contains the separated edge points can be potentially an improved input for some direct methods.

The same RGB images (Image 21), as in the previous implementation, are used in combination with the Canny edge detector. Two of the label channels, which are produced automatically using the "SemanticPass" script with Canny variation, are depicted in Image 26. It is worth noting that the produced point cloud contained 220.000 points while the detected edge points were 2.500.

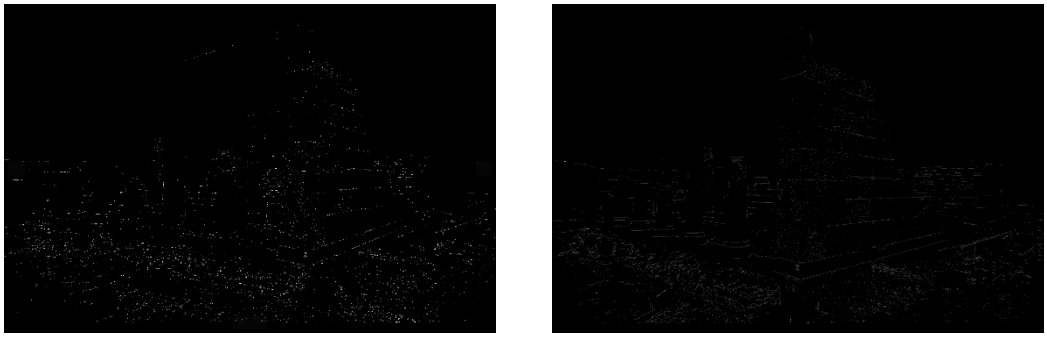
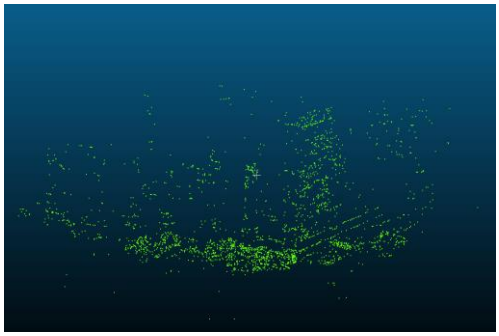


Image 26: Edge labels using Canny

The produced dense point cloud and the detected edges are depicted in Image 27.

Real colors point cloud Semantic colors point cloud



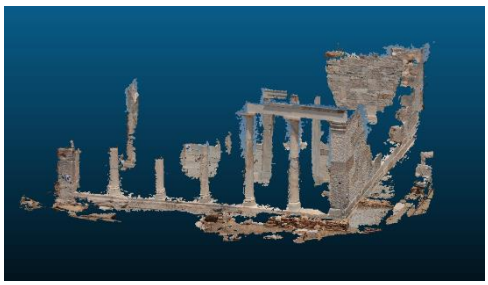
Extracted edges

Image 27: Detected edges using the OpenSfM and the Canny algorithm

3.4.3 Implementation using the Metashape software

Apart from the OpenSfM's SfM-MVS workflow, the Metashape software was also used. Agisoft Metashape is a closed commercial software and thus it was not possible to modify its source code. In fact, Metashape can process multispectral images by default. Thus, an implementation using the RGBL images was performed to evaluate its performance i.e., if the semantic information is inherited to the generated point clouds (sparse and dense). For the implementation using the Agisoft Metashape software, the Windows 10 operating system was used. In general, apart from the GUI, Agisoft provides a Python module from which it is able to manipulate the SfM-MVS procedure as well as the entire software's operations. In this diploma thesis two general approaches using the Agisoft Metashape software were implemented. The first one uses the standard GUI, while the second one uses the provided Python module. It is worth noting that the usage of the Python module, as well as the GUI, is allowed only for licensed users.

The first approach was conducted using the same images, which are depicted in Image 21, as in OpenSfM's implementation. The produced images, using the "SemanticPass" script, i.e., RGBL were added into the Metashape's chunk. They were then aligned, and the sparse point cloud was generated. Finally, using the sparse point cloud, the dense point cloud was produced. The sparse as well as the dense point clouds were extracted. The saved dense point cloud was further classified into edge points and all the rest using the labels. Finally, a line was produced via the points vectorization procedure, and saved using the ".dxf" format. The generated dense point cloud is depicted in Image 28, using two visualizations. The first one depicts the point cloud using the real colors i.e., RGB while the second one depicts the labeled points, with white, and all the rest, with black. In addition to the generated point cloud, the classified edge points, are also displayed in Image 28. Finally, the vectorized line is depicted in Image 29.



Real colors point cloud

Semantic colors point cloud

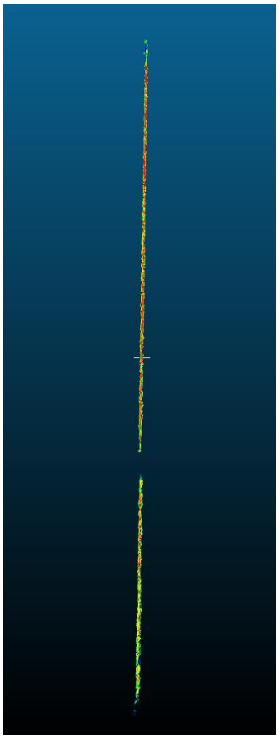


Image 28: Edge points detection using Agisoft-Metashape

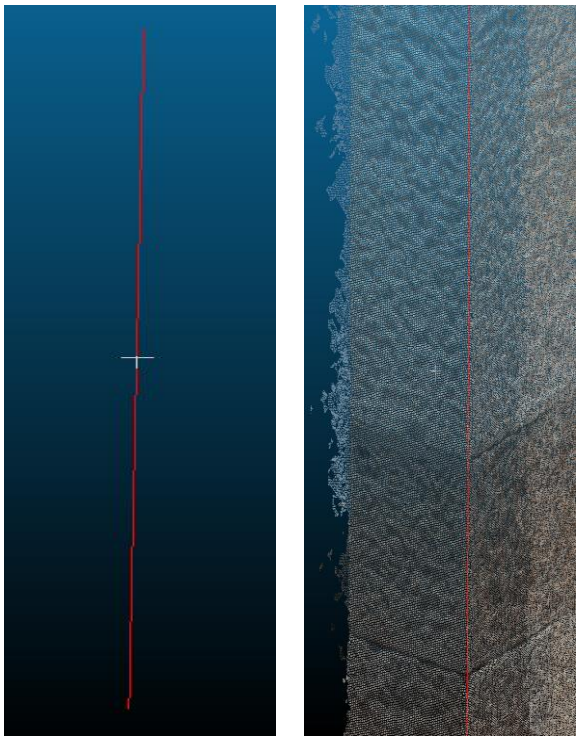
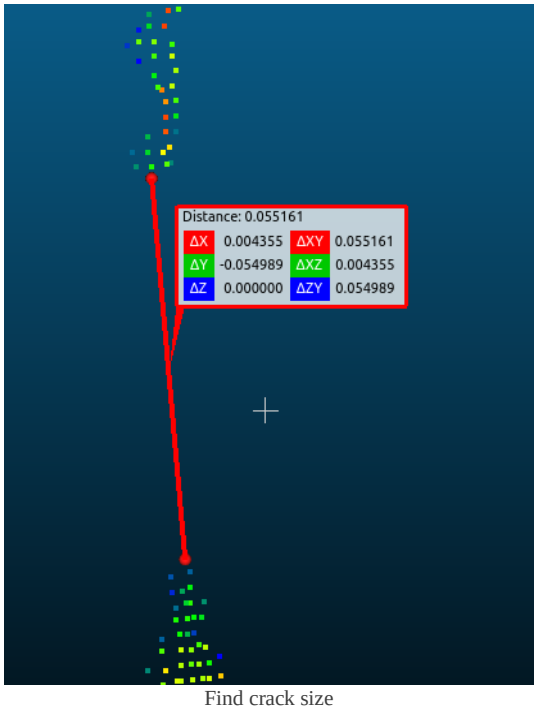


Image 29: 3D Line using Agisoft Metashape

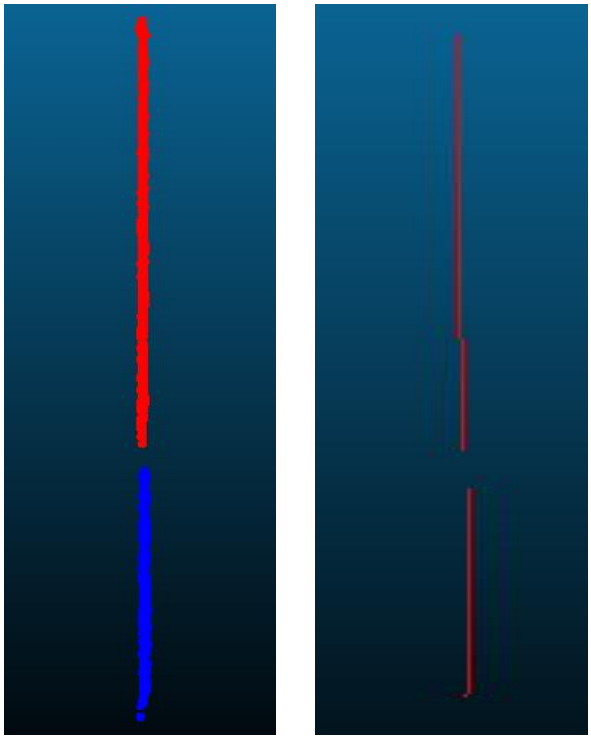
The generated point cloud was quite finer than the one produced with OpenSfM. In fact, only some noise points, from the sky were falsely generated. This issue can be solved using image masks which drop out the sky, or other unwanted areas during the procedure, but in this diploma thesis this image improvement was not implemented. The detected edge points are depicted in Image 28. The points which are displayed red are those with the highest label values while those which are depicted blue are those with the lowest label values. The red points constitute a more accurate representation of the 3D line because during the

process, even if the image was resized, their values were finally being near the original ones i.e., 255. Additionally, the detected edge points are better distributed than OpenSfM's implementation (Image 21). More concretely, all the detected points seem to lie on the same line. During the annotation procedure the line was drawn with 10 px width.

In fact, two lines constitute the annotated area due to a wall crack. The images were manually annotated considering this crack and thus, two lines were drawn. This can be also revealed from the detected points which can be decomposed into two classes. To achieve that, a further classification of the detected points must be performed. The classification process was conducted using a combination of the DBSCAN and RANSAC algorithms. Firstly, the DBSCAN algorithm is implemented to separate the detected points into two classes then the RANSAC algorithm is executed to classify the detected classes into inlier and outlier points. The inlier points constitute the set which is finally used for the vectorization task of each line. To execute the DBSCAN algorithm two parameters must be defined. The first one is the "eps" value from which each point's local area is determined. The "eps" value must be smaller than the crack's size, into the point cloud, to decompose the acquired edge point into two classes. To achieve that, the crack size was measured using the CloudCompare software (Image 30). The distance between the points was calculated as 0.05 units. Hence the eps value was defined as 0.04 units to be sure that an accurate separation will be performed. Finally, the vectorized lines are depicted in Image 30 as well as the separated points.



Find crack size



Detected classes Extracted vectors

Image 30: Further classification of the detected edge points

In fact, it is not efficient to measure each crack size into the produced point cloud to be able to detect them during the process. Thus, an estimated value according to the desired detail can be defined. The second parameter was the minimum samples value i.e., the minimum number of points which must contain a class, to be characterized as valid. The minimum samples value was defined as 2 because at least two points are sufficient to define a line. In this implementation the RANSAC algorithm is not necessary because noise edge points are not presented. The DBSCAN algorithm was selected to be used in this task, because the RANSAC algorithm due to its definition cannot identify holes according to the line direction. To be more specific, when two random candidate points are picked, and their candidate line is estimated, then each point distance is calculated. If the distance is smaller than a threshold value, the point is characterized as inlier otherwise as outlier. Thus, points from both categories will be classified as inliers during RANSAC execution, resulting to one entire class in comparison with the desired two. In contrast with RANSAC, DBSCAN uses the eps value to estimate each point's neighborhood. Hence, if the value is smaller than the crack size the two classes will be correctly separated, as the previous example presents. Finally, the "eps" value is estimated according to each implementation, considering image resolution as well as the dense cloud production performance.

Afterwards, an implementation using the Canny [Canny (1983) and (1986)] algorithm in combination with the Agisoft Metashape's SfM-MVS workflow was executed. The same images and edge semantic information, as the OpenSfM implementation, were used (RGB images (Image 21), Edge semantic information (Image 26)). The generated sparse and dense point clouds are depicted in Image 31 using two visualization approaches, respectively, one using the RGB colors and another using its label colors.

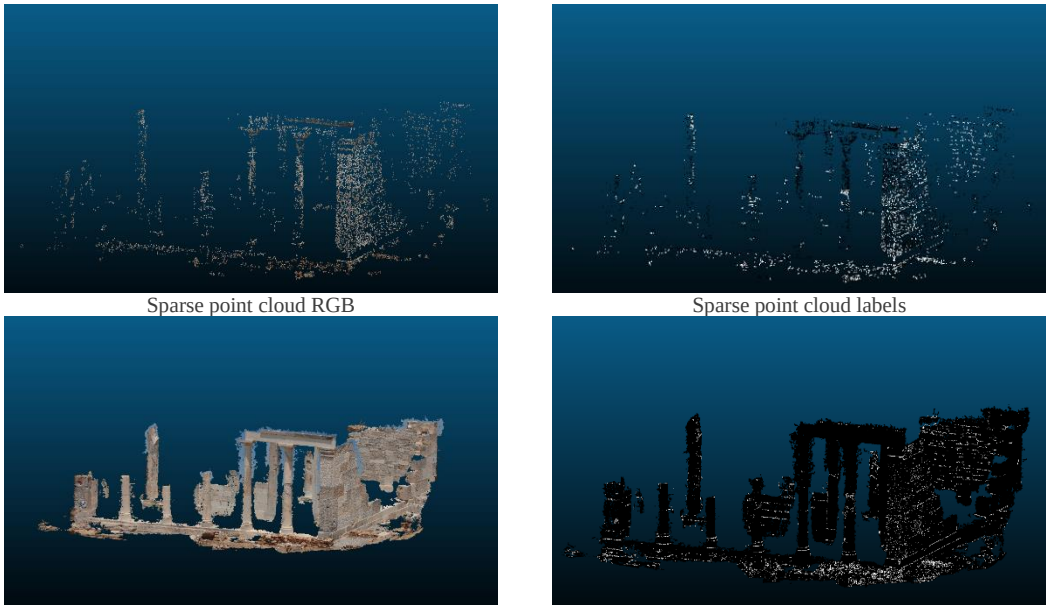


Image 31: Produced point clouds using Canny and Metashape GUI

The sparse point cloud contains 8.385 points from which the 2.543 are labeled as edge points i.e., 30%. The dense point cloud contains 5.919.302 points from which the 265.572 are labeled as edge points i.e., 4% while those with values over 100 are only 43.283 points i.e., 0.7%. First and foremost, the Canny [Canny (1983) and (1986)] algorithm did not detect accurately all the building edges. Those comparisons are performed to highlight the contribution of the edge points (features) into the SfM procedure. The RGB sparse and dense point clouds are the same as the previous ones (Image 26) as long as the same images and settings were selected during the SfM-MVS workflow. Afterwards, the edge points were extracted in the same way as for the previous implementations. The detected edges are depicted in Image 32.

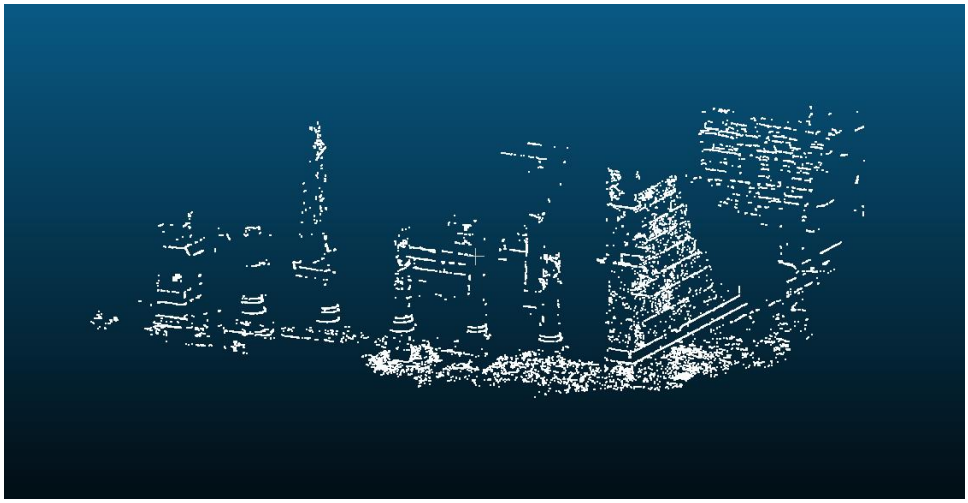


Image 32: Detected edge points using Canny and Metashape GUI

The detected edge points were further analyzed using the previously described classification procedure i.e., via the implementation of DBSCAN and RANSAC algorithms. The classified regions are depicted in Image 33.

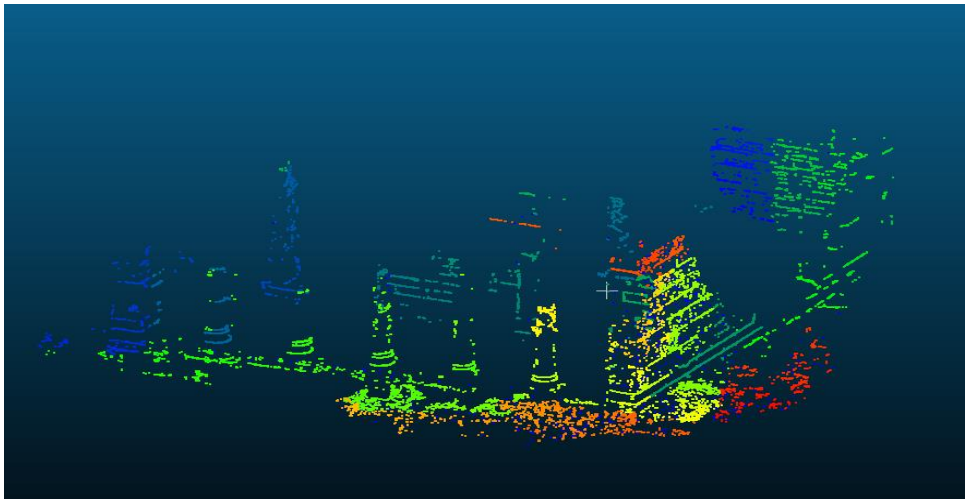


Image 33: Detected lines' classes using Metashape GUI

A lot of colors which are contained in Image 33, seem to be the same. In fact, they are slightly different i.e., different classes, as the extracted 3D vectors in Image 33 indicate. The detected noise points are depicted in Image 34.

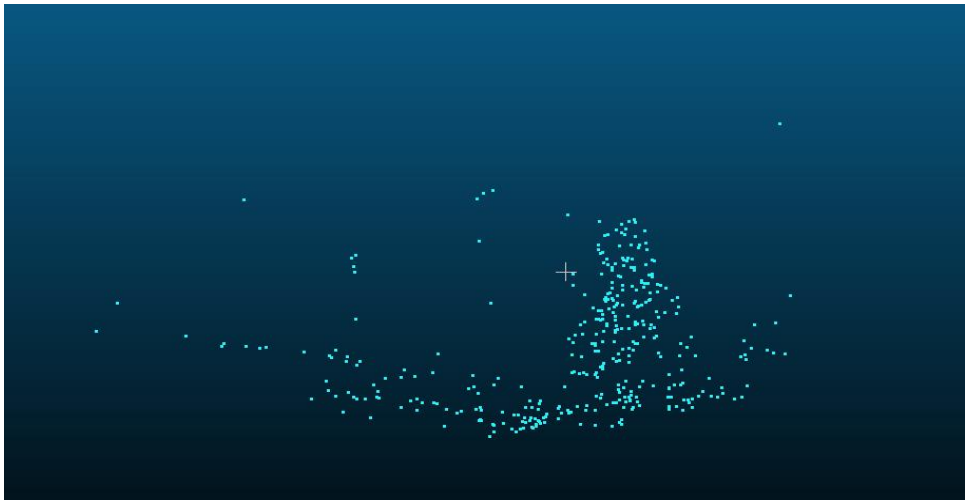
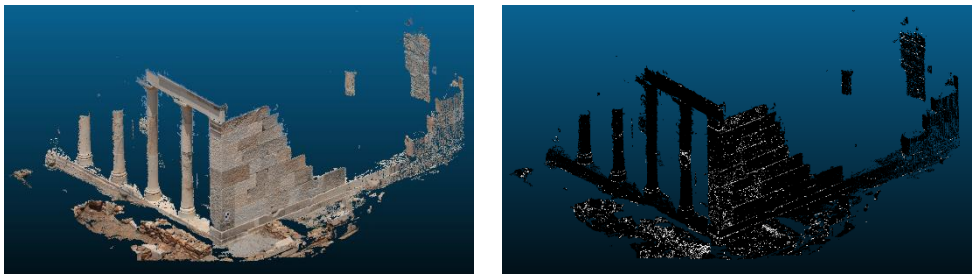


Image 34: Noise points using Metashape GUI

Finally, the extracted vectors are presented in Image 35. It is worth noting that, the extracted vectors were produced with a random implementation. More concretely, once the detected edge vectors are classified into the areas, which are depicted in Image 33, then the first and the last point of each class are selected. Those points are used to define the 3D vector. Then each vector is added into a ".dxf" archive. The 3D vectors are produced only to present a general method which must be improved as well as to be further investigated.

Image 35: 3D vectors using Canny and Metashape GUI

Besides the implementation which uses the Metashape GUI, an implementation using Metashape's Python module was performed. Thus, a new script called "Metashape_SFM.py" was developed in which a class called "MetaSFM" was created. The produced class takes the project name using the ".psx" format as input, e.g., "project.psx". Then it reads the images which were produced by the "SemanticPass" script. In fact, these images are the enriched images i.e., RGBL. Afterwards, a function called "sfmmvs" is executed. This function manipulates the entire SfM-MVS process using the provided module called "Metashape". The first step into the "MetaSFM" class is the creation of the "Metashape.Document()" in which all the operations are stored. Then a new chunk is created and the RGBL images are read and stored. Afterwards, the image matching and camera aligning procedures are performed. Hence, the sparse point cloud is generated. Then, a depth map for each image is created. Finally, the dense point cloud is calculated, exploiting the SfM procedure as well as the calculated depth maps. Once the dense cloud is produced, then it is stored as "merged.ply". The generated dense point cloud is used for further analysis of edge detection and vectorization in the same way as it was conducted in the previous steps. It is worth noting that, "Metashape" python module provides a plethora of classes and functions, with a variety of variables, which can be changed and modified according to the dependencies of each implementation. An implementation using three images from the ancient temple of Demeter dataset, as well as the Canny algorithm, was performed. The produced dense point cloud is presented in Image 36. The detected edges are displayed in Image 37 while the detected line classes are presented in Image 38. The noise points are depicted in Image 39. Finally, the vectorized 3D lines are presented in Image 40.



Dense point cloud RGB Dense point cloud labels

Image 36: Produced point clouds using Canny and Agisoft Metashape python module

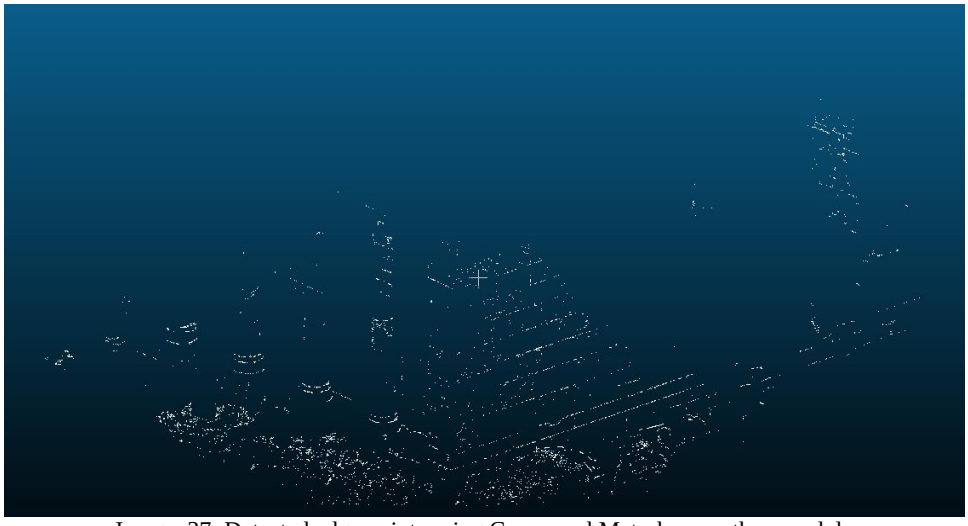


Image 37: Detected edge points using Canny and Metashape python module

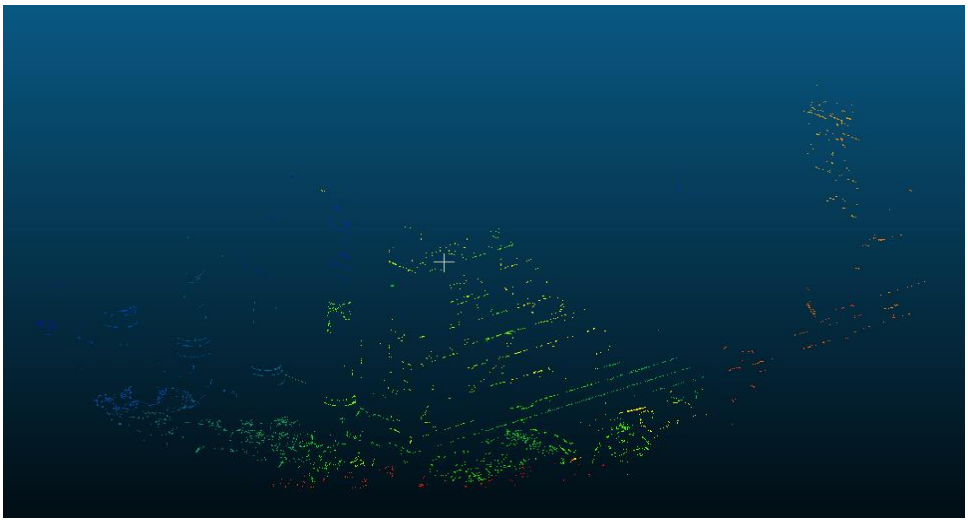


Image 38: Detected lines' classes using Metashape python module

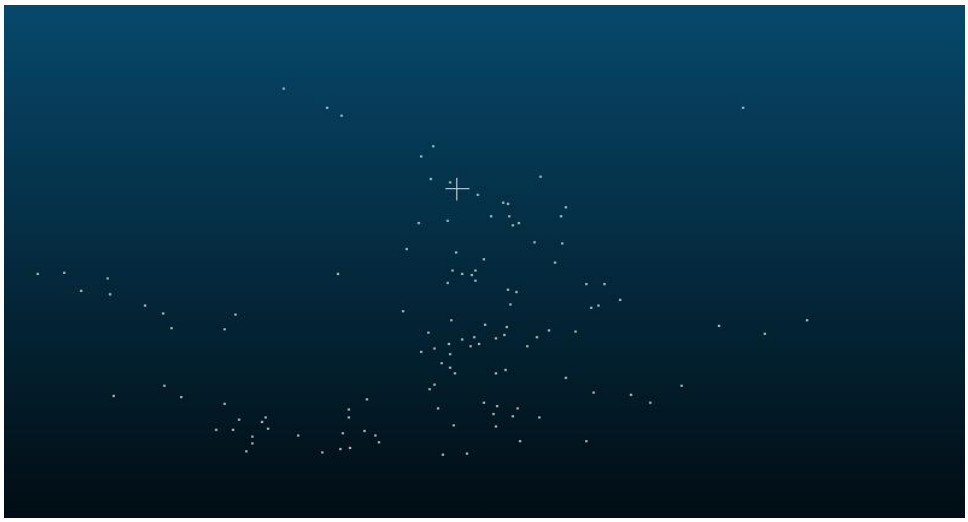


Image 39: Noise points using Metashape python module

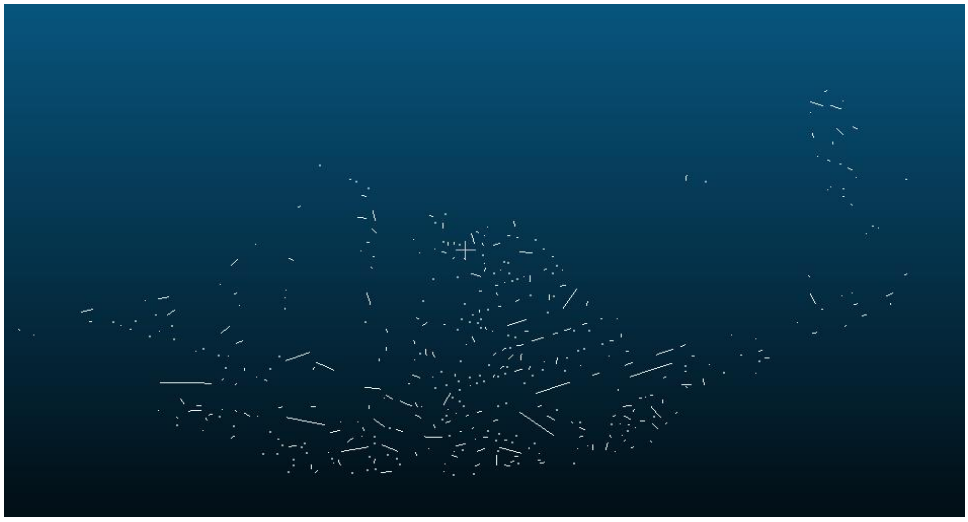


Image 40: 3D vectors using Canny and Metashape python module

3.5 Proposed Approach

The proposed approach combines the developed approaches into a script called “3DPlan”. Additionally, improves the 2D edge detection task, introducing a live Canny implementation inspired by Arapellis (2020). The user interacts with the “3DPlan” script, to define the procedure’s necessary parameters. Additionally, the “3DPlan” script provides the automated implementations of the developed algorithms. The general structure of the proposed approach is depicted in Image 3.

Firstly, the user defines if the produced labeled images, which will be imported into the SfM-MVS procedure, will contain 3 or 4 channels. The 4-channels approach is the recommended one because it has the benefit to keep the real image colors during the SfM-MVS procedure and so to construct a point cloud which contains the real point colors. The detected edges can be visualized using a software which can manipulate multichannel images or selecting an option into the “3DPlan.py” workflow which enables labels export. In contrast with the 4-channels approach the 3-channels one provides a direct visualization of the detected edges, because the red channel is replaced with the edge semantic information and thus the detected edges are displayed red, into the produced images.

Secondly, the user selects the algorithm which will be used to generate the semantic point cloud. Three options are provided, (i) the Agisoft Metashape, (ii) the Mapillary OpenSfM and (iii) the MyTriangulation software. The Agisoft Metashape choice is separated into two branches the (i) Python Module and the (ii) GUI. The user selects the branch. Independently to the selection i.e., GUI or Module a directory, called “Lines”, is constructed into the “3DPlan” directory. Then if the user’s choice was the Python Module, the script called “Metashape_SfM” is executed. The produced dense point cloud i.e., merged.ply, is stored into the “Lines” directory. If the user’s choice was the GUI, the “3DPlan” script stops temporarily its execution, until the dense point cloud be generated, using Agisoft Metashape GUI. When the dense point cloud is produced, must be manually stored as “merged.txt”, and the user moves it into the “Lines” directory. The Mapillary OpenSfM choice firstly creates the environment in which the algorithm will be executed. To be more specific, the working directory is changed from the “3DPlan” to “OpenSfM” installation directory. Then a folder into OpenSfM “data” directory is automatically constructed, called “3DPlan”. The created directory consists the working directory for the execution of the OpenSfM algorithm. In fact, is the corresponding directory, according to this implementation, to the “berlin” one from the first implementation. Afterwards, a directory called “images” is automatically created into the “3DPlan” one and the enriched images i.e., RGBL, are automatically copied into it. Besides the RGBL images the “config.yaml” archive is automatically copied from the “berlin” directory to the “3DPlan” one. At this end, the entire necessary environment to execute the OpenSfM workflow is

ready. The entire process is manipulated using the “osfm_env” function from the “utils” script. Then, the “3DPlan” script executes the command “/OpenSfM/bin/opensfm_run_all” using the terminal. Hence, the OpenSfM workflow is executed and the merged.ply is produced and copied into the “Lines” directory. The MyTriangulation choice firstly creates, in the same way as the previous choices do, the “Lines” directory into the “3DPlan” one. Then executes the “MyTriangulation” workflow as it was described in the developed approaches.

Thirdly, the user selects the method which will be used to define the edge semantic information. Two approaches are provided, the first one is the (i) live Canny implementation while the second one is the (ii) external semantic source. The live edge detection uses the Canny algorithm and is implemented using two windows. The first window displays the original image or “MyImage” while the second one displays the dynamically changed edge map. Additionally, two bars from which the two variables i.e., min and max can be modified during the process, are provided. The user moves the given bars to define the minimum and maximum values, which are used into Canny’s hysteresis thresholding step. The set of the values which accomplish the best results, according to the user’s perspective, exploiting the live edge map window, are selected. The edge map is stored, as the edge semantic information for the under-process image using an additional or the red channel. Then the next image is imported into the live edge detection procedure. The external source choice provides the ability to pre-define a directory which contains the edge semantic information for each image. This is provided, because it gives the opportunity to the user to exploit more accurate edge semantic information than Canny, e.g., manually annotated images, if it is possible. The user must construct a directory called “semantic_images” in which the semantic information for each image must be stored, using the “[image’s name]_l.jpg” format (image’s name must not contain, RGB image’s suffix).

Fourthly, the user defines the given RGB images format. The RGB images are stored into the “rgb” folder which is into the “3DPlan” directory. A plethora of available formats are provided for the RGB images, like ‘.JPG’, ‘.jpg’ and ‘.tiff’.

Afterwards, the “SemanticPass.py” script is executed. According to the selected method of the acquisition of the edge semantic information, i.e. “Canny” or “external source”, the “SemanticPass.py” script is executed differently. If the external source choice is selected, the under process RGB and semantic, images are traversed from its directory i.e., “rgb” and “semantic_images”. If the semantic images consist of three channels, they are further analyzed to extract the semantic information into a label channel. Otherwise, the given one channel image is directly associated with the under-process image’s label channel. Afterwards, the enriched image is produced, combining the RGB channels with the labeled one. Then, it is saved into the “images” folder. This procedure is executed iteratively for each RGB image. If the Canny choice is selected, two windows are opened automatically, enabling the live process for each image in the “rgb” folder. The two windows are depicted in Image 41. This process was added into the workflow to handle Canny’s drawback, which is the dependence of the definition of its variables on each image and not to the depicted objects, by adjusting the min and max variables using the provided toolbars. The live editing procedure is manipulated by four functions, the (i) define_min, the (ii) define_max, the (iii) new_value and the (iv) find_edges. The first one assigns into a dictionary, which is used as the values of Canny’s parameters, the minimum threshold value while the second one, executes the same operations but for the maximum threshold value. The third one updates the minimum and the maximum values using user’s input. Finally, the last function applies the Canny algorithm using the dictionary’s parameters. In fact, the entire process is dynamically changed according to user’s inputs thus the previous functions are executed iteratively in respect to the, each time, new parameters.

Image 41: Canny's windows

When the user is satisfied about the depicted edge map, the process moves forward to the next image by pressing the 'Q' button. Then, the RGBL images are constructed and saved into the "images" folder. The improved "SemanticPass" script provides the opportunity to blur the RGB images, using a Gaussian kernel, before the execution of the Canny algorithm. The user can define the kernel's size as well as can save the blurred images into the "Blur_rgb" folder which is constructed automatically. Additionally, the improved "SemanticPass" script gives to the user the ability to extract the produced edge maps, using the live Canny implementation, into a folder called "Labels".

The produced RGBL images are fed into the selected SfM-MVS algorithm or the "MyTriangulation" workflow, and the semantic point cloud is produced and stored into the "Lines" directory as "merged.txt" or "merged.ply". Then a function called "classify_points", which aims to separate the labeled points from all the rest, is executed. This function reads the enriched point cloud, line by line and identifies if the label value is over a pre-defined threshold value or not. By default, the threshold value is defined as 100 but the user may modify this value. The definition of the threshold value is conducted according to which edge points the user wishes to be extracted e.g., only the "strong" edge points or all the labeled points etc. It is worth noting that, apart from the explanation that the label values are changed during the process because the images are resized one more may exist. Each pixel in each image is associated with a label. If an edge is detected in some images but in some others, in which it is also depicted, is not detected, the label value to the produced 3D point will perhaps be changed from 255, according to the color manipulation procedure. To be more specific, if the color determination is performed by calculating for each point the mean color value of the pixels from which is produced, then the same calculation will be performed for the label values and thus the points' labels will not be equal to 255. The detected points are extracted into the "edges.txt" file. Finally, the detected points are imported into the "Clustering" script which aims to further classify the detected edge points into n-separated set of points which represent an edge i.e., to decompose the detected edge points into each edge's points. This clustering procedure is performed using the DBSCAN algorithm while the implementation was described previously. The produced set of points i.e., clusters, are further analyzed using the RANSAC algorithm. The inlier points which are produced via RANSAC algorithm are used for the vectorization task, in which a simplified method is used. To be more specific, the first and the last inliers are extracted and the line which is defined using them is vectorized. Finally, the created vectors are saved into the "3DPlan.dxf" archive. The vectorization task is handled by the "lines2dxf" function which is defined into the "utils" script.

The entire process of the "3DPlan" script as well as the interaction with the user was constructed with a "defensive" perspective. To be more specific, some functions were constructed to handle the entire interaction in a way to predict common user mistakes as well as to provide informative messages during the process and when issues occur. To this end the functions "message", "error_message" and "input_check" were created into the "utils" script. The first function is an alternative to Python "print"

function, but in addition to the printed text the “message” function provides the information of the date and time of its execution. The “error_message” function is used when an issue occurs. It has two different approaches. The first one just raises a message, as the function “message” do. The second one firstly prints the error message and then terminates the process. The third function i.e., “input_check”, interacts with the user. It takes three arguments as input, the message, the valid answer and the error message. The message prints to the user the question e.g., “Agisoft-Metashape (0), Mapillary-OpenSFM (1) or MyTriangulation (2)? (Write 0, 1 or 2):”, the “valid answers”, is a list of the expected user’s answers i.e., [0, 1, 2], while the error message is raised if the user’s answer is not the proper one, without terminating the procedure. Some messages and errors during the process are depicted in Image 42, as examples of the previously described approach.

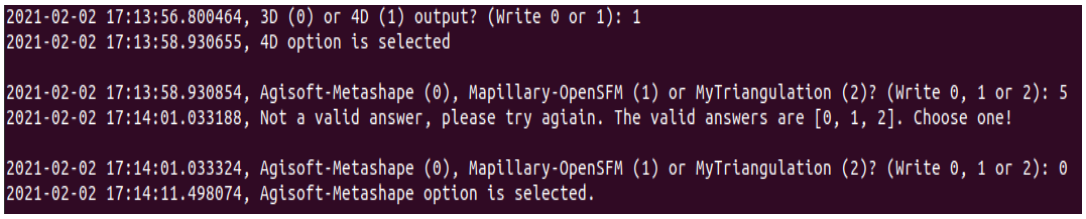


Image 42: Messages and errors using the 3DPlan script

The flowchart of the proposed approach is depicted in Figure 4.

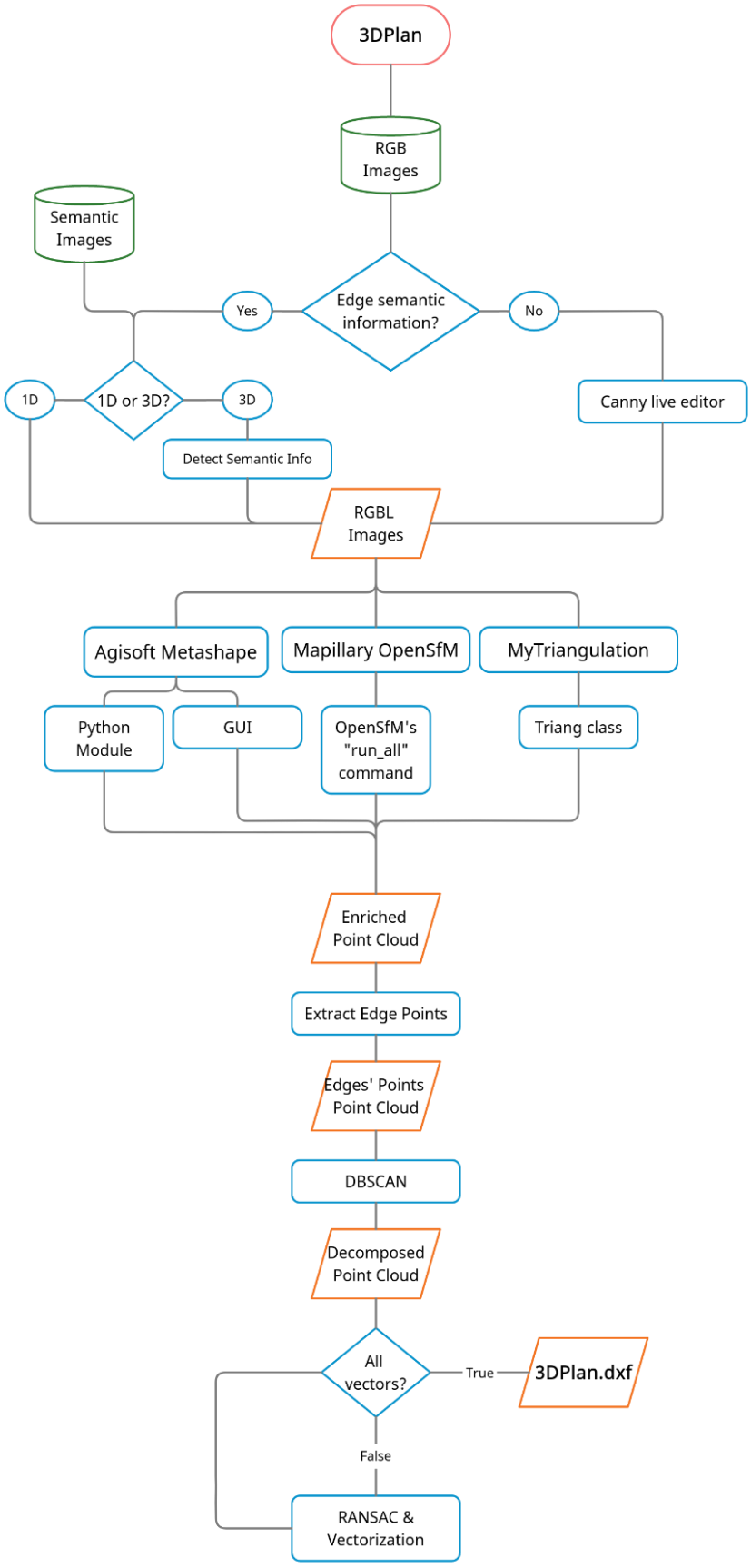


Figure 4: 3DPlan flowchart

The Figure 5 displays the directory's structure. This visualization aims to present the folders' position in respect to the 3DPlan directory. A potential user must set the directories in the same way as the Figure 5 depicts, to construct the valid structure for the 3DPlan execution, especially when the OpenSfM implementation is used.

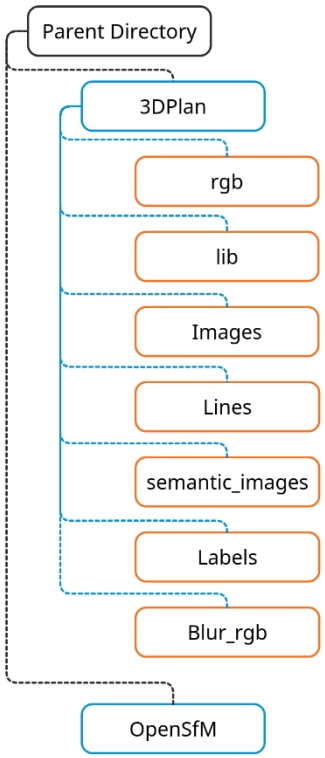
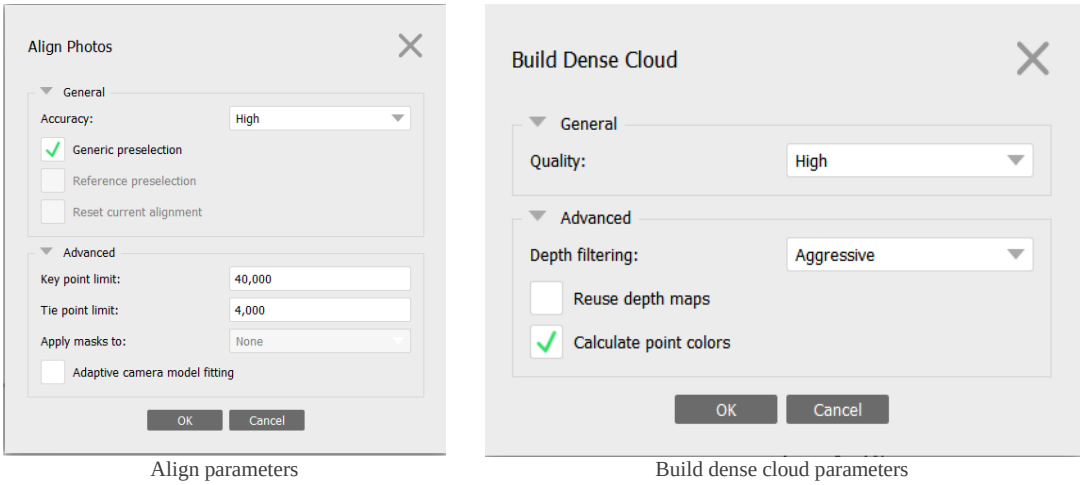


Figure 5: Directory's structure

The displayed objects in Figure 5 are directories. The "rgb" directory contains the given RGB images. The "lib" directory contains the previously described scripts e.g., "SemanticPass", "Clustering" etc. The "3DPlan" script is included into the "3DPlan" directory but is not depicted in this figure because it is not a directory. The "Images" directory stores the enriched images i.e., RGBL. The "Lines" directory contains the generated products e.g., "3DPlan.dxf", the enriched point cloud etc. The "semantic_images" stores the edge semantic information, if it already exists, as 1D or 3D image, for each RGB image i.e., the "external source" approach. Instead, the "Labels" one contains the edge semantic information which is produced using the live Canny editor i.e., each RGB image's edge map. Additionally, the "Blur_rgb" folder stores the blurred images which can be produced before the execution of the live Canny editor. These smoothed RGB images, resulting to better edge maps than the original because the textured areas are smoothed as it was described previously. Finally, the directory in which the OpenSfM algorithm was installed is depicted as "OpenSfM" directory. It is worth noting that the "OpenSfM" directory is included in the same directory as the "3DPlan" one i.e., under the "Parent Directory". The folders' structure is a crucial detail which leads to a successful implementation.

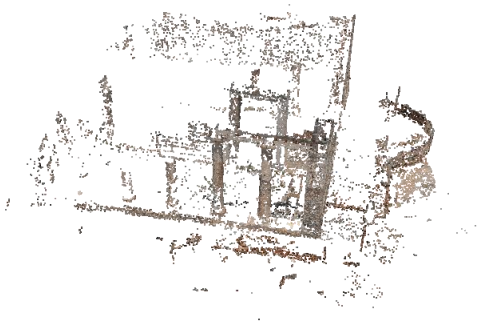
Afterwards, two implementations exploiting the proposed approach with the live-Canny variation, were conducted using the Temple of Demeter and the Police Station datasets. In fact, each implementation contains sub-implementations in which the DBSCAN's "eps" value and the RANSAC's threshold value, are changed. The products for each implementation are the same i.e., the sparse and dense point clouds, the detected edges, the lines classes and the vectorized lines. The sparse and the dense point clouds are the same for the sub-implementations of each dataset. Thus, they are depicted in Image 44 and Image 49, respectively. Additionally, the detected edges were the same as long as the same enriched dense point

cloud was used. Thus, the detected edges are also depicted in Image 45 and Image 50, respectively. From the available SfM-MVS workflows, the Agisoft Metashape software was selected due to its high performance. The production of the sparse and dense point cloud of the Temple of Demeter was conducted using 49 images while the production of the Police Station ones was conducted using 43 images. The align parameters as well as the parameters which were used for the dense cloud production are depicted in Image 43.

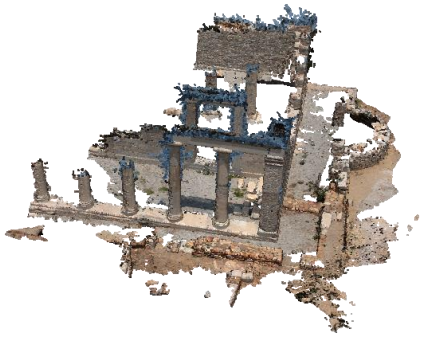


Align parameters Build dense cloud parameters

Image 43: Align and build dense cloud parameters



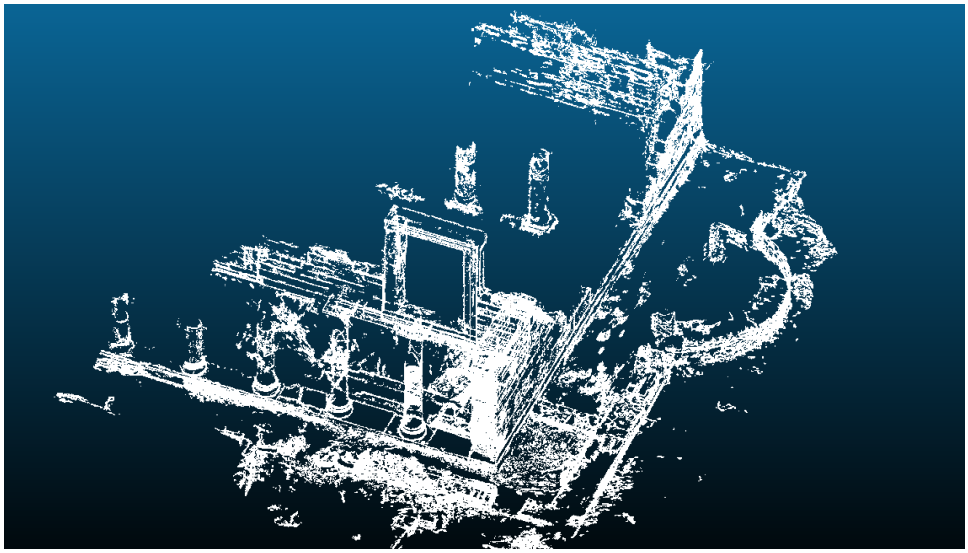
Sparse point cloud RGB



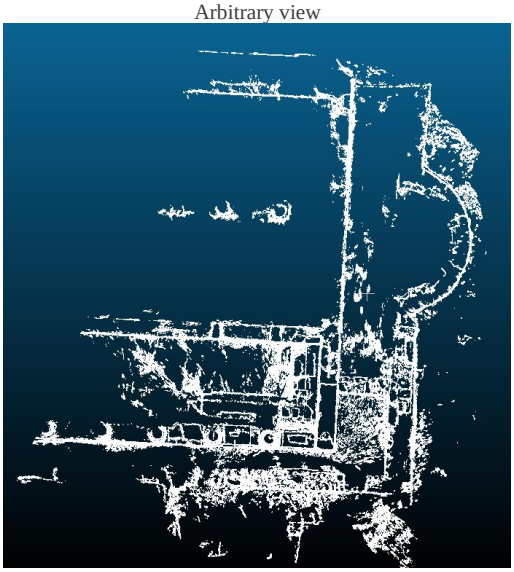
Dense point cloud RGB

Image 44: Proposed approach Temple of Demeter point clouds

The edges were extracted exploiting their label value, as it was described in the previous sections, and are presented in Image 45.



Arbitrary view



Bird's eye view

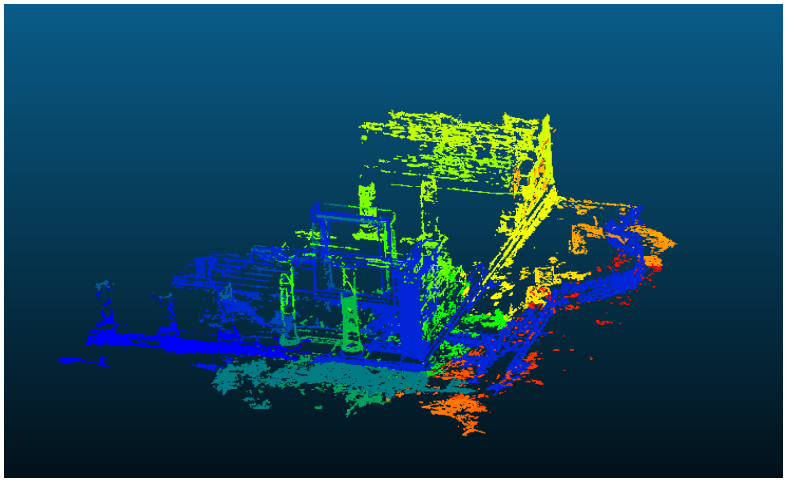
Image 45: Detected edges Temple of Demeter

The parameters, which were used for each sub-implementation are depicted in Table 2.

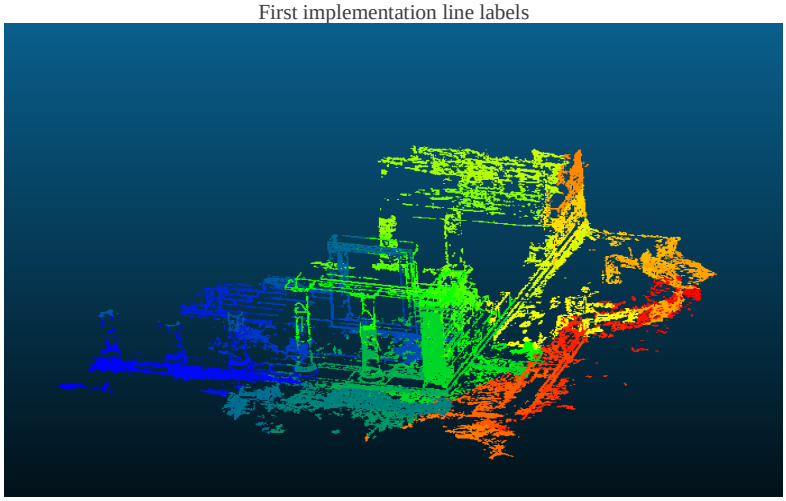
Table 2: Proposed approach Temple of Demeter parameters

Implementation	eps (units)	Minimum samples	RANSAC threshold (units)
1	0.01	10	0.009
2	0.005	10	0.004

The calculated line classes are depicted in Image 46 while the detected noise points are presented in Image 47. Finally, Image 48 displays the vectorized lines in combination with the sparse point cloud as well as with different view angles, to be more intuitive.

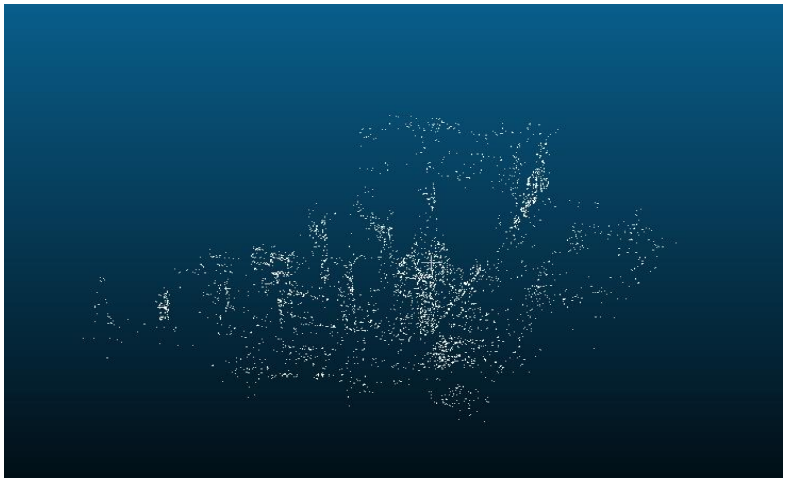


First implementation line labels

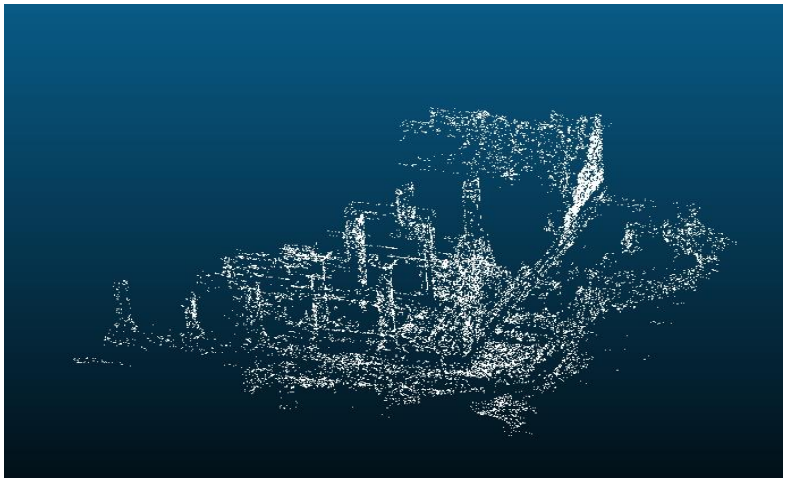


Second implementation line labels

Image 46: Line classes Temple of Demeter

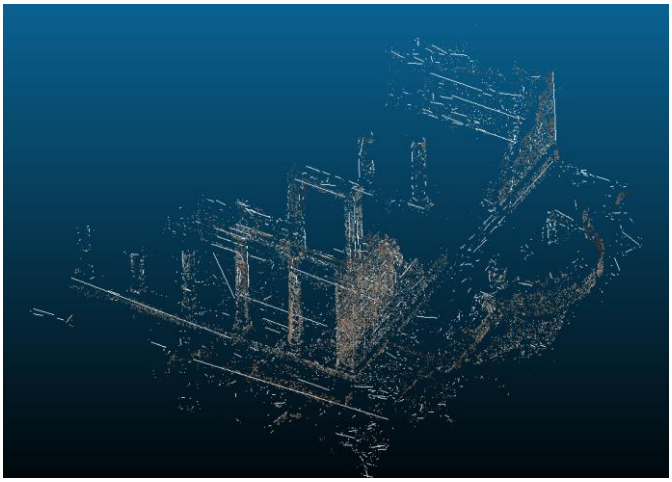


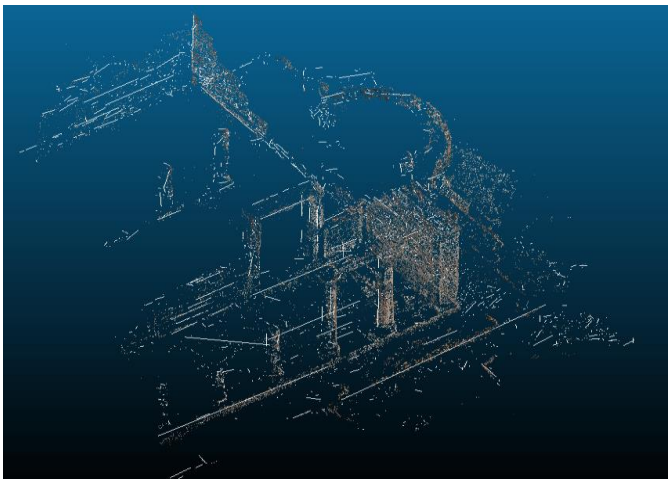
First implementation noise points

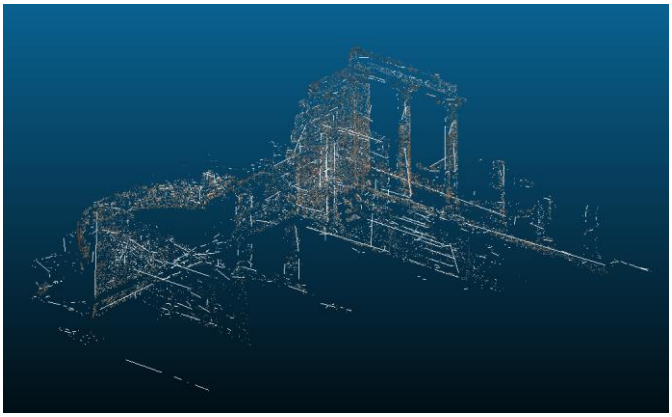


Second implementation noise points

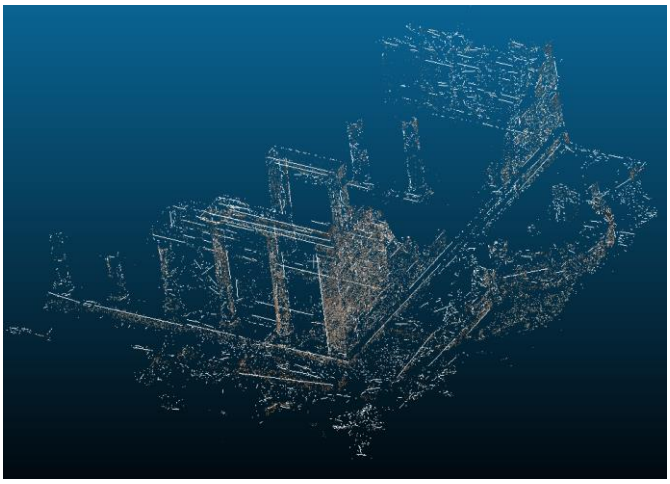
Image 47: Noise points Temple of Demeter

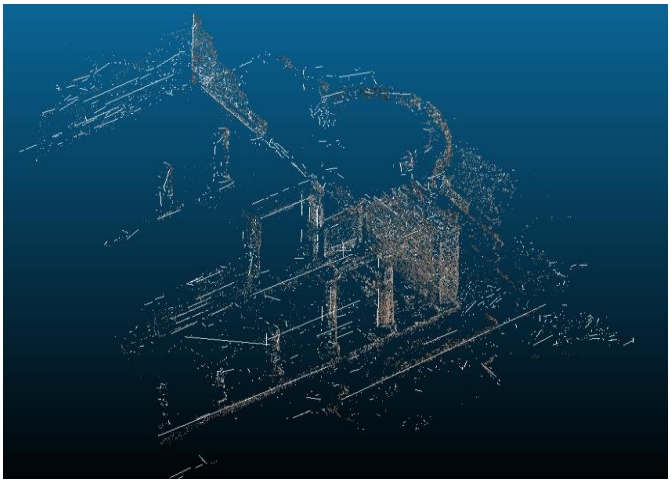


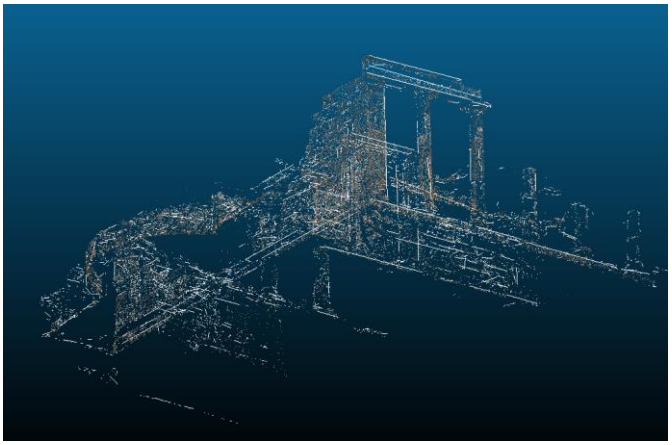




First implementation vectorized lines

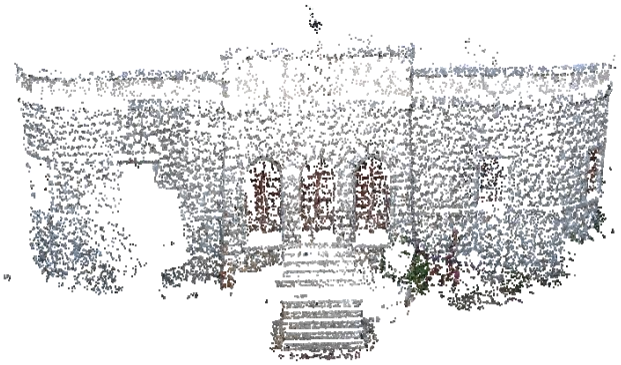






Second implementation vectorized lines

Image 48: Vectorized lines with sparse point cloud Temple of Demeter

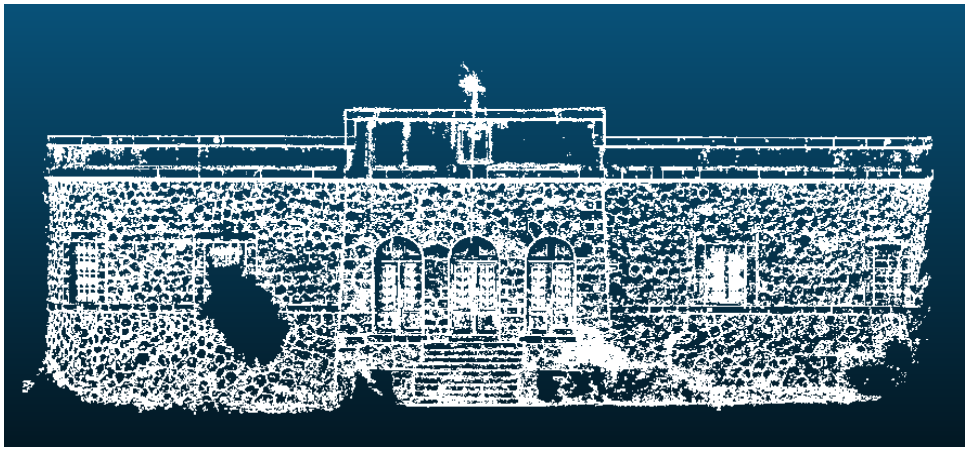


Sparse point cloud RGB

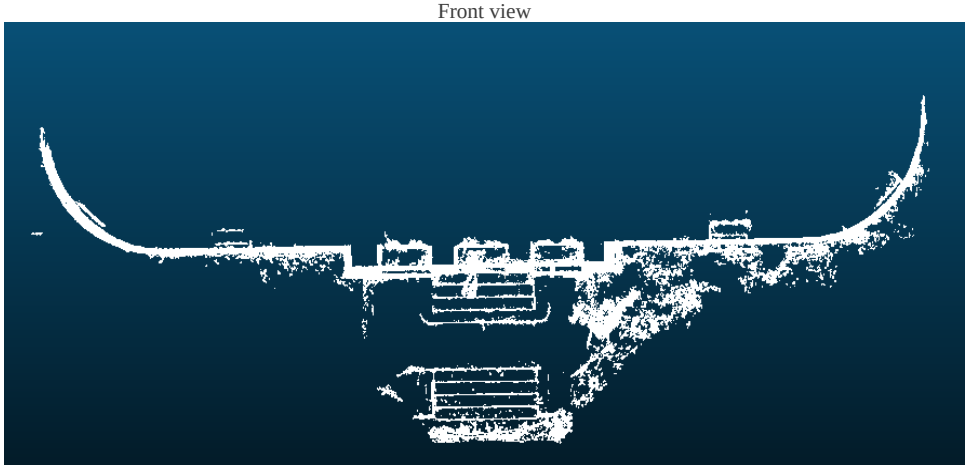
Dense point cloud RGB

Image 49: Proposed approach Police station point clouds

The edges were extracted exploiting theirs label value, as it was described into the previous sections, and are presented into Image 50.



Front view



Bird's eye view

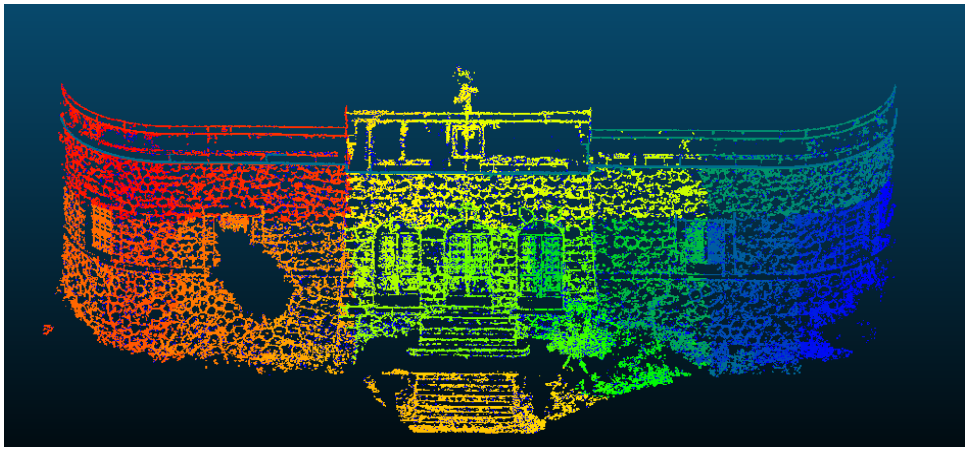
Image 50: Detected edges Police Station

The parameters, which were used, for each sub-implementation are depicted in Table 3.

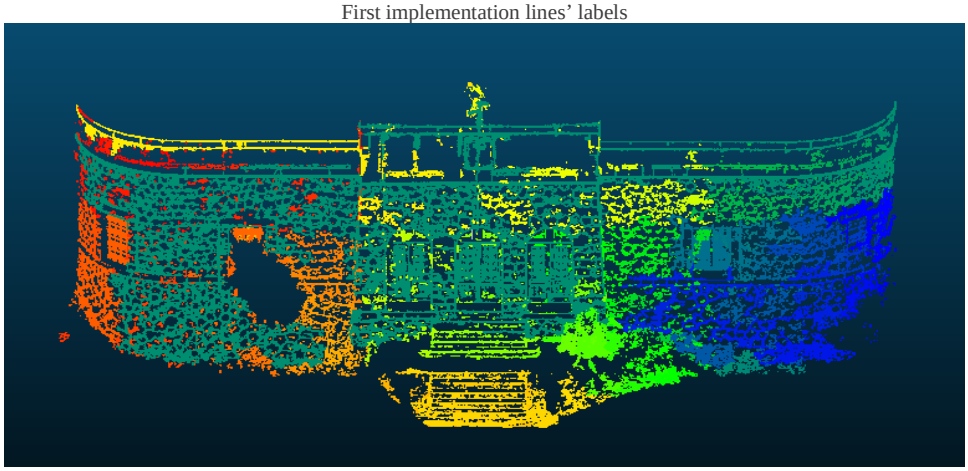
Table 3: Proposed approach Police Station parameters

Implementation	eps (units)	Minimum samples	RANSAC's threshold (units)
1	0.01	10	0.009
2	0.03	10	0.029

The calculated line classes are depicted in Image 51 while the detected noise points are presented in Image 52. Finally, Image 53 displays the vectorized lines in combination with the sparse point cloud as well as with different view angles, to be more intuitive.



First implementation lines' labels



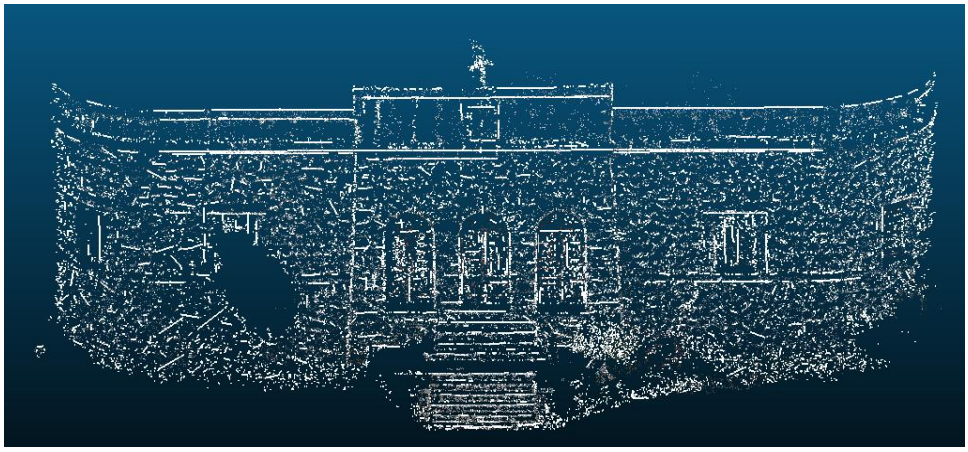
Second implementation lines' labels

Image 51: Line classes police station

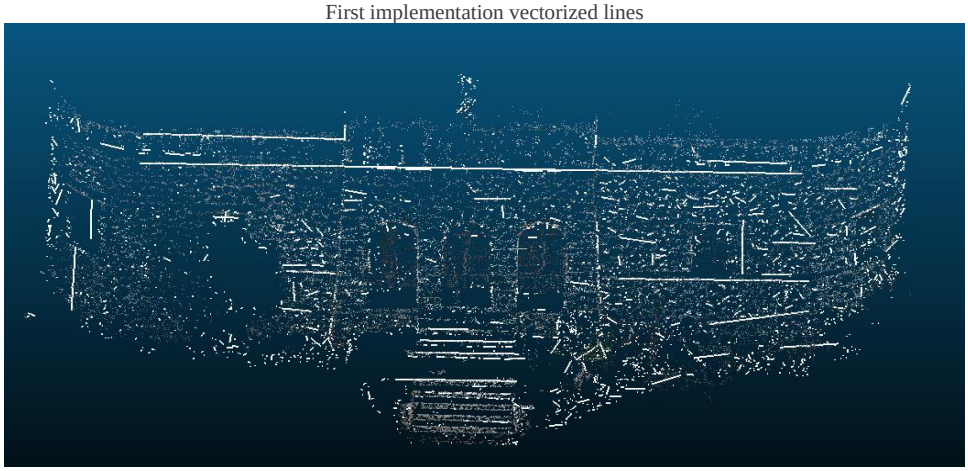
First implementation noise points

Second implementation noise points

Image 52: Noise points police station



First implementation vectorized lines



Second implementation vectorized lines

Image 53: Vectorized lines with sparse point cloud police station

References:

Agisoft Metashape (v.1.5.2) (2021) *Agisoft Software* [Online]. Available at: https://www.agisoft.com/ (Accessed: 08 January 2021)

Arapellis, O. 2020. Semiautomated edge detection on digital images. Postgraduate Degree Thesis, Postgraduate Course in Geoinformatics, NTUA (in Greek).

Alcantarilla, P.F. and Solutions, T., 2011. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, *34*(7), pp.1281-1298.

Bay, H., Tuytelaars, T. and Van Gool, L., 2006, May. Surf: Speeded up robust features. In European conference on computer vision (pp. 404-417). Springer, Berlin, Heidelberg.

CloudCompare (version 2.11.1 (Anoia)) [GPL software]. (2021). Retrieved from http://www.cloudcompare.org/ (Accessed: 09 January 2021)

Daniel Girardeau-Montaut. Détection de changement sur des données géométriques tridimensionnelles. domainother. Telécom ParisTech, 2006. English.<pastel-00001745> (20) (PDF) Détection de changement sur des données géométriques tridimensionnelles. Available from: https://www.researchgate.net/publication/29973519_Detection_de_changement_sur_des_donnees_geometriques_tridimensionnelles [accessed Feb 11 2021].

Git (version 2.30.1). (2021). Available at: https://git-scm.com/ (Accessed: 14 January 2021)

Giannakoula, X., 2018. Geometrical Documentation of monuments using modern technologies, an application on the Temple of Demeter in Naxos. (In Greek)

Lowe D.G. 2004 Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision 60(2), 91–110

Meshroom (version 2020.1.0) (2021) *Meshroom Software* [Online]. Available at: https://alicevision.org/#meshroom (Accessed: 09 January 2021)

Muja, M. and Lowe, D., 2009. Flann-fast library for approximate nearest neighbors user manual. Computer Science Department, University of British Columbia, Vancouver, BC, Canada.

National Archeological Museum (2021), Archaic Period [online]. Available at: https://www.namuseum.gr/collection/archaiki-periodos-2/ (Accessed: 06 January 2021) (In Greek)

OpenSfM source code (version 0.5.1) [BSD-2-Clause software] (2021), *OpenSfM source code* [Online]. Available at: https://github.com/mapillary/OpenSfM (Accessed: 09 January 2021)

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011, November. ORB: An efficient alternative to SIFT or SURF. In 2011 International conference on computer vision (pp. 2564-2571). IEEE.

Stefanakis, M.I., Kalogeropoulos, K., Georgopoulos, A. and Bourbou, C., 2015. 10 Exploring the Ancient Demos of Kymissaleis on Rhodes: Multidisciplinary Experimental Research and Theoretical Issues. Classical Archaeology in Context: Theory and Practice in Excavation in the Greek World, pp.259-314.

Stefanou A. B., 2018. The contribution of new technologies to archeology: the case of the classic church of Demeter in Sagri Naxos. (In Greek)

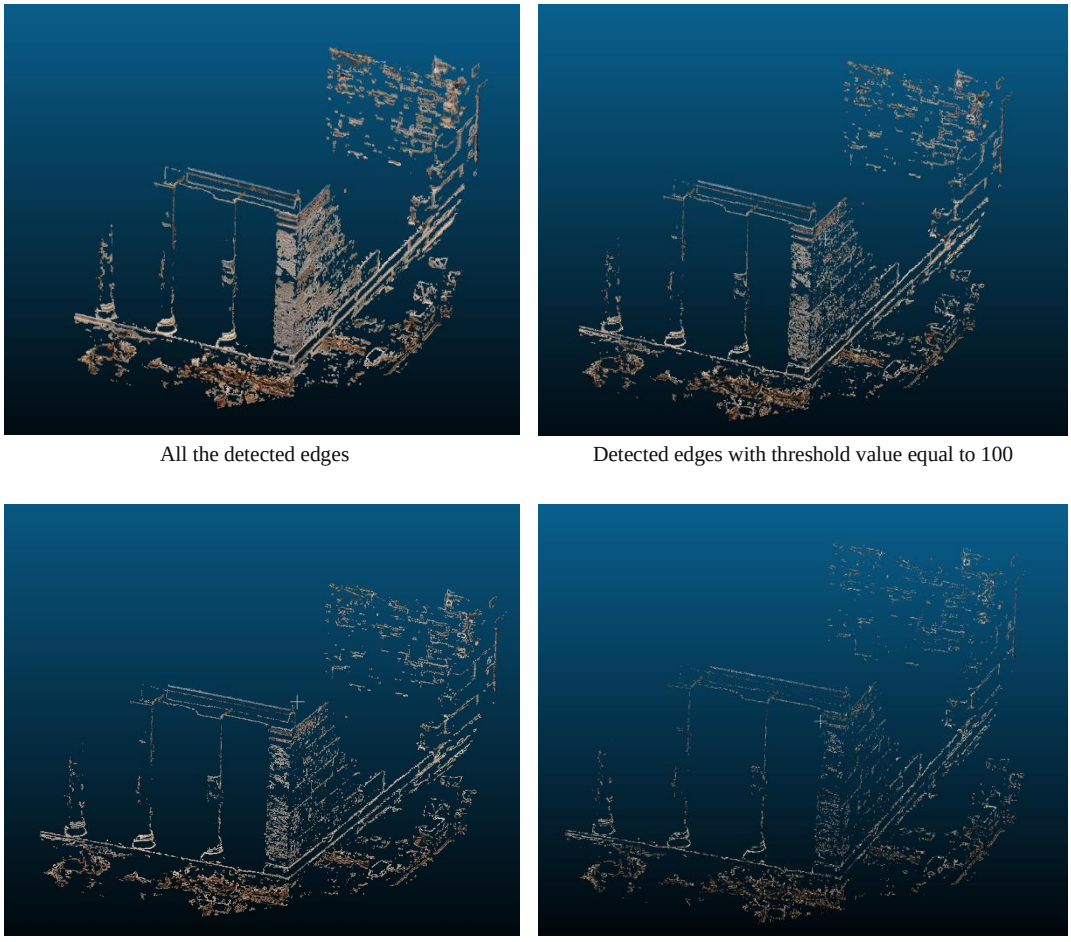
4. Conclusions

The aim of this diploma thesis was to extract 3D edges from point clouds exploiting edge semantic information. The evaluation of the proposed approach is based on the assessment of the individual steps of the algorithm i.e., the edge semantic information production, the dense point cloud creation, the chosen parameters for the clustering step, as well as the vectorization procedure. First and foremost, the quality of the edge semantic information, highly influences the extraction of each edge. The proposed approach provides two methods for the acquisition of the edge semantic information i.e., using the "live Canny" or the "external source", approach. The first approach, opens two windows automatically, enabling the live process, in which the user, according to his perspective, produces the edge map for each of the given images. On the other hand, the "external source" approach, offers the opportunity to detect the edges via other edge detectors or to manually annotate them on the used images. Hence, the proposed approach can easily be combined with external edge detection scripts, saving the produced edge maps into the "semantic_images" directory. Thus, if a potential user wants to apply his edge detection techniques, the execution procedure could not be changed. The quality of the produced edge semantic information can be evaluated comparing the detected edges with the real ones. To this end, a comparison between them was conducted using a set of 4D images, which were produced by the "3DPlan" script with the live Canny approach, as well as the QGIS software. The images carry the edge semantic information as a 4^{th} channel. Hence, the images were opened using the QGIS software and visualized replacing the red channel with the 4^{th} one. An area of a produced image is depicted in Figure 1.

Figure 1: Canny edge detection visualization

The detected edges suffer from misdetections especially in the textured areas. This drawback is partially managed by the live-Canny detection approach, but it is not completely eliminated. In fact, the Canny edge detector extracts the areas in which the illumination is changed drastically and thus the textured areas lead to noise, i.e., useless edges. Apart from that, misdetections occur due to the shadows present. Thus, the user could schedule the time of capture to avoid them, where possible. The detected edges are

not imported directly into the vectorization step. Hence, misdetections can be excluded during the classification task, which aims to classify the points into edge points and all the rest, as well as during the clustering task, which aims to decompose the entire set of edge points into several groups, representing each of the object's edges. The produced point cloud i.e., "merged.txt" or "merged.ply" contains at least seven values for each 3D point. The position x, y and z the colors r, g and b and the label value l. The classification procedure reads each registration of the produced point cloud and selects the points with label value over or equal to a given threshold value. The pixel values of the 4th channel are 255 for the edge points and 0 for all the rest. Thus, the desired label value, in the point cloud archive, is 255. However, several values between 0 and 255, are presented as labels. In fact, during the SfM-MVS procedure the images are resized, to reduce the computational cost of the workflow. Therefore, the interpolation method, which is used to resize the images, mixes each label value with its neighborhood ones. Additionally, some edges are not labeled in all images in which they are depicted, due to Canny weaknesses. Hence, if the SfM-MVS software uses the labels with the same way as the colors, i.e., finds a mean value of the colors or labels from all the pixels which are associated with the desired point, the lack of the label information i.e., 0, leading to change the value from 255 to a lower one. To this end, a threshold value is selected e.g., 100 to separate the edges to "strong" and "weak". The "strong" edges are fed into the clustering task. In general, the erroneous edges are much smaller than the valid ones. To avoid the misdetections, the "min_samples" variable, into the DBSCAN algorithm, could be defined as a larger number e.g., 10, although the minimum number of points to define a line is two. Thus, the smaller lines will be classified as noise due to their neighborhood which will not contain the necessary number of points. In combination with the "eps" value, which must be smaller than the average distance between the generated points, the spurious detections can be eliminated during the classification task, resulting to a more accurate 3D plan. Further investigation to improve the detected edge points should be performed, although several approaches, using different "eps" values as well as "min_samples" ones, were conducted during the practical implementation. Additionally, some real edges are not detected during the procedure and thus they are not included into the final product. Unfortunately, those real edges are excluded during the live edge detection task, simultaneously with some erroneous edges. An improvement of the detection procedure could be a post process of the given images e.g., enhancing them using a bilateral filtering technique, to prepare them for the edge detection task. Such a method could be included into the "Semantic_Pass" script, before the live-Canny detection. In the provided script a blurring procedure is included, using a Gaussian kernel. Thus, this procedure can be improved using a biliteral filter or other methods to detect the edges with the most feasible accuracy. The extraction of the 3D edges is strongly associated with the 2D edge detection procedure i.e., the edge semantic information, and thus a further investigation of the traditional as well as the Machine Learning or Deep Learning (ML-DL) techniques could be performed. To sum up, three filtering techniques are applied on the labeled points: (i) the threshold label value, (ii) the DBSCAN algorithm and the (iii) RANSAC. The remaining points of the first process are depicted in Figure 2.



All the detected edges

Detected edges with threshold value equal to 100

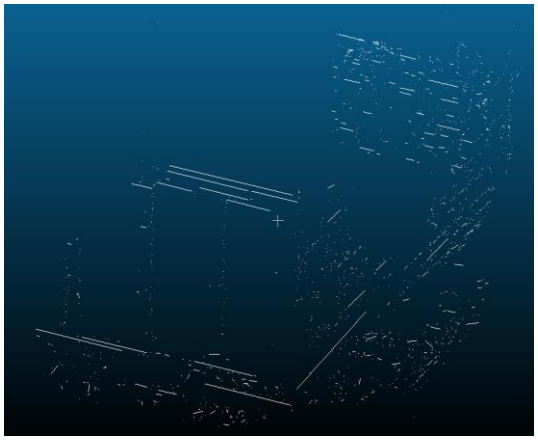
Detected edges with threshold value equal to 150

Detected edges with threshold value equal to 200

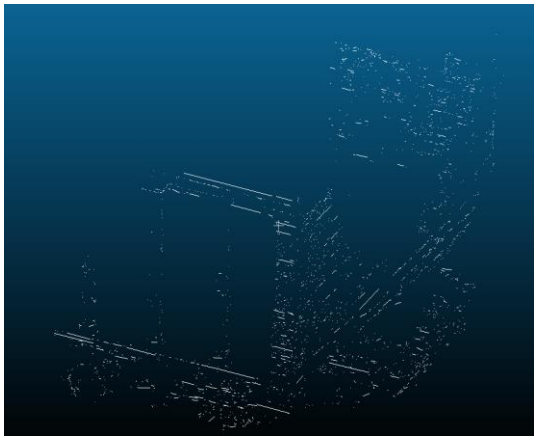
Figure 2: Detected edges with several label threshold values

Apart from the edge semantic information, the performance of the SfM-MVS workflow plays a significant role to the product's quality. The point cloud must be dense and accurate. To be more specific, an accurate point cloud i.e., with reduced noise and without lack of information e.g., holes, as well as a point cloud with a dense distribution of points, gives the opportunity to the user to identify the edges with the most feasible accuracy. To this end, the Agisoft Metashape as well as the OpenSfM software were connected with the proposed algorithm to execute the SfM-MVS procedure. Metashape provides a guaranteed SfM-MVS workflow which is used in several photogrammetric implementations however it is a "black box". Instead, the OpenSfM license does not guarantee the output results, but an experienced user can modify the workflow to each application dependencies as well as to validate the generated point cloud. In fact, the OpenSfM approach expects professional users i.e., with an in-depth knowledge of the SfM-MVS procedure in comparison with the Agisoft Metashape which can be used by inexperienced ones, because it provides a user-friendly graphical interface. More concretely, the OpenSfM contains a lot of parameters which should be manually defined to achieve the desired results. In this diploma thesis most of the parameters were not changed and thus the produced point cloud was not the most accurate one, as this was not the main aim of the thesis. In fact, the dense distribution of points provides the necessary information for the vectorization step. A common issue during the production of point clouds, is the presence of blue points, close to the edges, due to the sky. Thus, the images should be masked before the execution to achieve better results. Additionally, the parameters into the clustering task i.e., "eps", "min_samples" and "RANSAC threshold" are very important. These parameters are used into the DBSCAN and the RANSAC algorithms. The sets of points, which will be vectorized, are created with respect to those parameters. Furthermore, those parameters define the accuracy of the detected edges. To

be more specific, each point's neighborhood can be extracted using several "eps" values. Before the execution of the clustering step, a preprocess of the produced point cloud could be performed. More precisely, the points which refer to the ground could be excluded to improve the vectorization step. Assume that for a given point two implementations have been performed. In the first experiment the "eps" value is equal to 0.01 while in the second one it is equal to 0.001. The units of the "eps" value are the same as the units of the generated point cloud. In this diploma thesis the point cloud has an arbitrary coordinate system and thus the "eps" value's units are considered as "units" e.g., 0.01 units etc. The first implementation produces a set of points in which each point is at most 0.01 units away from at least one neighbouring point, while in the second one each point is at most 0.001 units away from at least one neighbouring point. Thus, if an accurate and dense point cloud exists in which the acquired points are no more than 0.001 units far away, the first implementation will return a better set of points for the vectorization task. To be more specific, once the edge extraction as well as the clustering procedures, are performed, each group is fed into the vectorization step. In this diploma thesis the vectorization step is performed using the first and the last point of the extracted set. This approach is not efficient and thus new approaches must be included. The produced set of points contain a dense distribution of points which describe an object edge. Thus, the entire information i.e., all the detected points, must be exploited, during the vectorization step to produce an accurate line. This can be performed using the parameters, which are estimated by RANSAC during the clustering procedure. In fact, the RANSAC algorithm apart from the inlier points, estimates each line's parameters. Hence, those parameters could be exploited to predict the line's points and then to vectorize them. In general, the included parameters are strongly associated to each other as well as to the case study object. Thus, a manual definition of them i.e. directly into the "3DPlan" script, is recommended. An example of the vectorization accuracy is presented in Figure 3.



3D plan using all the labeled edges



3D plan using the edges with labels >= 100

3D plan using the edges with labels >= 150

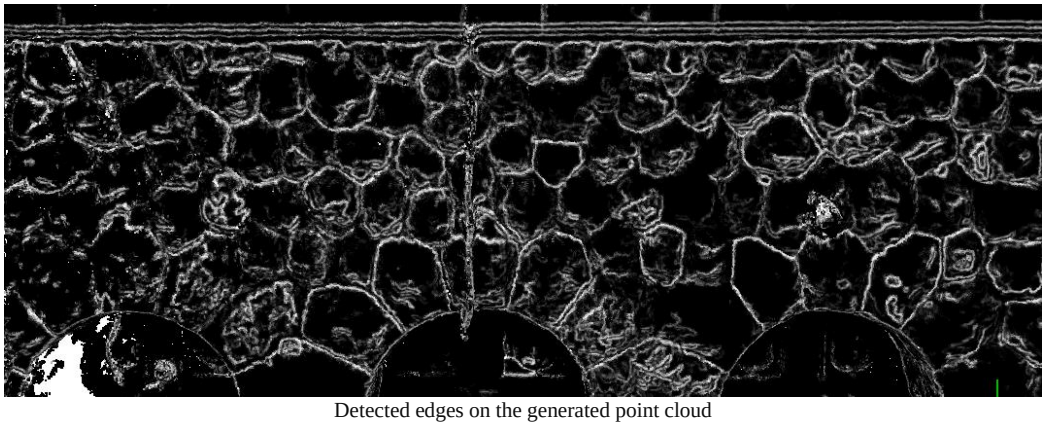
3D plan using the edges with labels >= 200

Figure 3: 3D plan with several label threshold values

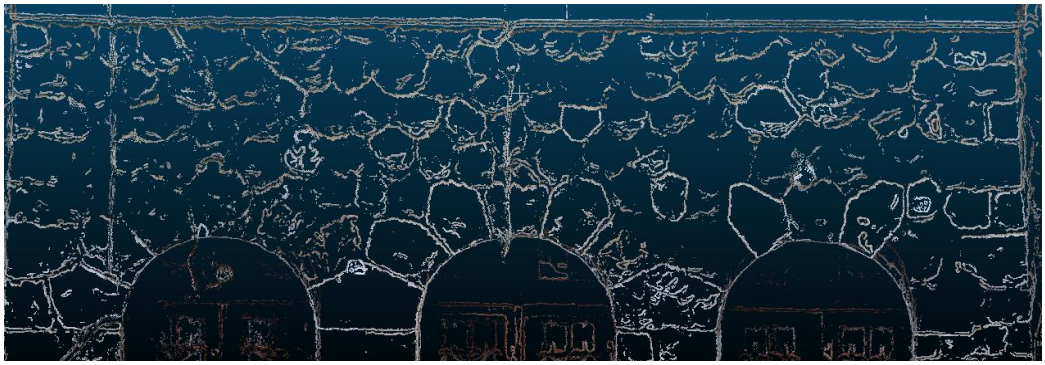
It is worth noting that the straight lines which are depicted in Figure 3.1, are decomposed to the next implementations, i.e., 3.2, 3.3 and 3.4, into smaller ones. This happens due to the edge point's upsampling using the different threshold values. In fact, the other parameters i.e., "eps", "min_samples" and "RANSAC threshold", remain the same, passing from the one implementation to the other to investigate the effect of the threshold value to the entire procedure.

Besides the algorithm's steps, each case study object, affects the quality of the final product. To be more specific, a complex object decreases the final product quality due to the existing small edges which must be detected, extracted and vectorized. For example, the police station is an object built of stone and thus a lot of small edges which describe each stone are presented. These edges are similar with the noise edges, produced due to the textured areas e.g., on the stones or on textured wall (red parallelogram). Thus, this complex geometry e.g., each stone, is very difficult to be detected and separated from noise, during the process. An example of the detected edges, using the Canny edge detector, on an arbitrary image of the police station dataset is depicted in Figure 4 while a view of the created dense cloud as well as the generated 3D edges, in Figure 5.

Figure 4: Stone wall edge detection



Detected edges on the generated point cloud



Detected edges with threshold value equal to 100

Figure 5: The dense cloud and the 3D detected edges

Several misdetections occur, especially on the stones, despite the fact that a vast number of edges, depicted in Figure 5, are valid. Additionally, using the threshold value, apart from the noise, valid edges are also eliminated. Thus, several approaches could be conducted to find the best threshold value depending on each case study object. Furthermore, a comparison between the vectorized lines with the corresponding real ones should be implemented. To be more specific, the proposed approach could be performed using a georeferenced point cloud as well as conventional geodetic measurements or the conventional photogrammetric process. Then, the vectorized lines could be compared with the real ones, providing useful metrics as well as several conclusions about the accuracy of the proposed approach. It is worth noting that, extra mathematical equations are not used during the implementation of the proposed approach, apart from the conventional dense cloud production workflow, such as planes' intersection, and thus additional errors are not included.

The proposed algorithm is executed using the script called "3DPlan". Firstly, the user defines several variables, answering the provided questions. Thus, the necessary conditions under which the algorithm will be executed, are defined. For example, the user defines the SfM-MVS algorithm, which will be used, the given images' suffix etc. There are not a lot of questions during the process as well as all the necessary ones are displayed immediately when the script is executed. Hence, the user should not be aware of answering questions during the process. Additionally, a "defensive" approach, which is described into the practical implementation section, is developed in which a potentially "wrong" answer is not leading to termination of the entire procedure as long as it is not necessary. The proposed interaction approach is not the most efficient one. An alternative is the usage of an "argument parser" approach in which the necessary variables will be defined with the execution command e.g., using the command "python3 3DPlan.py --4D --Metashape --GUI --Canny --.JPG" for an execution, exploiting the Metashape GUI software, using ".JPG" images, the "Canny" variation for the edge semantic information production and 4D images creation. This approach improves the proposed algorithm, and it is the best one for a software, which will be used with console commands. In fact, the "argument parser" approach eliminates the users who are not familiar with Python and thus the best interaction way among the users and the provided software is a graphical user interface (GUI), due to its simplicity. One of the future additions to improve the efficiency of the software developed would be to build a "professional" GUI for the inexperienced users. Finally, it is worth noting that the OpenSfM approach can be executed only using Linux, but the Metashape one can be used in any operational system.