Πανεπιστήμιο Πατρών
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Μιχαήλ Μαναγούδης 1059398

Υπολογιστική Γεωμετρία και Εφαρμογές 3Δ Μοντελοποίησης

Απαλλακτική Εργασία : **2.Multimodal**

Διευκρινήσεις:

- ***Οι τεχνικές που αναλύονται παρακάτω και αντίστοιχα τα αποτελέσματα των οποίων αποτυπώνονται στις εικόνες εφαρμόστηκαν σε πολλαπλές εισόδους (2D εικόνων και lidar εικόνων)***
- ***Οι παράμετροι και τα όρια μεταβλητών που χρειάζονται διάφορες συναρτήσεις τέθηκαν κατόπιν πειραματισμού, μεταξύ πολλών εναλλακτικών τιμών, με στόχο την βελτίωση την απόδοσης του προγράμματος.***

1. Image-space lane detection

a. Carla

i. Grayscale.

Αφού φορτώσουμε την έγχρωμη εικόνα έπειτα την μετατρέπουμε σε grayscale με την κατάλληλη συνάρτηση της OpenCV (cvtColor).

Έγχρωμη εικόνα

Ασπρόμαυρη εικόνα

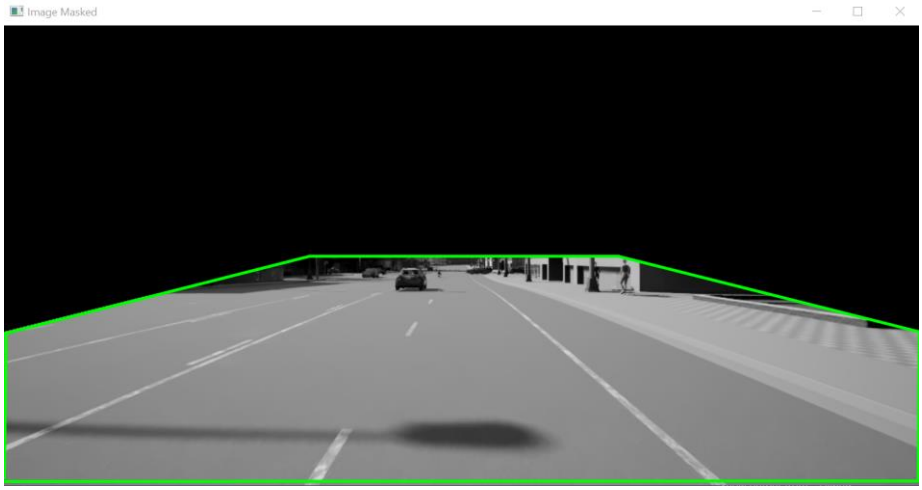
ii. Denoising.

Αντίστοιχα με την συνάρτηση fastNlMeansDenoising() αφαιρούμε αποτελεσματικά τον θόρυβο από την εικόνα.
Προαιρετικά μπορούμε να εφαρμόσουμε και κάποιο φίλτρο στην εικόνα (πχ Gaussian blur).

Αρχική εικόνα

Αποθορυβοποιημένη εικόνα

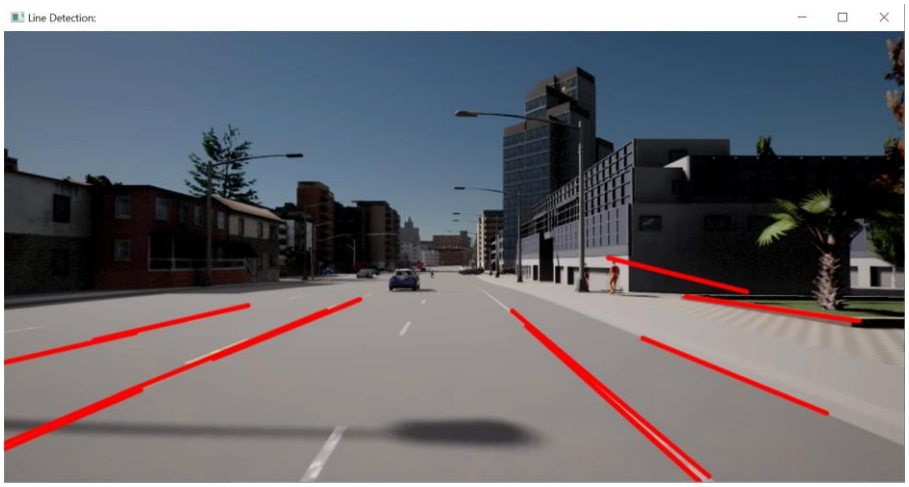
Επιπλέον εφαρμόζουμε μία μάσκα στην εικόνα για να ασχοληθούμε με την περιοχή ενδιαφέροντος (που είναι συνήθως στο κέντρο και προς τα κάτω).



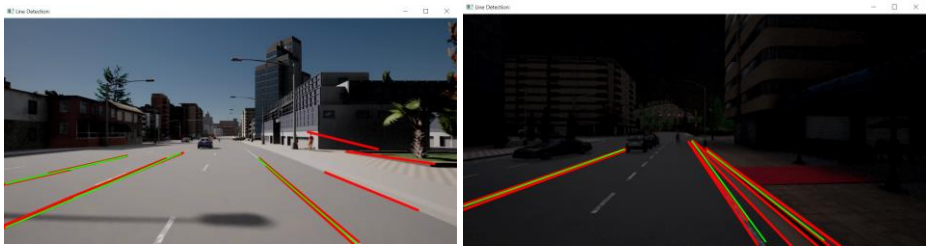
iii. Edge & lane detection.

Με την μέθοδο Canny edge detection (και τις κατάλληλες παραμέτρους που υπολογίστηκαν έπειτα από διαδοχικές δοκιμές) ανιχνεύουμε τις κυριότερες ακμές στην εικόνα. Να σημειωθεί ότι η συνάρτηση Canny της OpenCV χρησιμοποιεί και ένα φίλτρο Sobel εσωτερικά της.

Έπειτα εφαρμόζουμε τον Probabilistic Hough Line Transform (καλύτερος από τον απλό Hough Line Transform αφού μεταξύ άλλων υπολογίζει τις γραμμές ως ευθύγραμμα τμήματα και όχι ως ευθείες) για την ανίχνευση των γραμμών του δρόμου.



Στην συνέχεια χρειάζεται ένα ακόμη βήμα για την ολοκληρωμένη ανίχνευση γραμμών του δρόμου το οποίο δεν έχει προλάβει να συμπεριληφθεί στον κώδικα. Το βήμα αυτό είναι μία ομαδοποίηση των γραμμών (παρόμοια τεχνική ομαδοποίησης υλοποιείται στο 2^{ο} μέρος της εργασίας lidar-space) . Συγκεκριμένα όπως φαίνεται και στις προηγούμενες εικόνες για την ίδια γραμμή του δρόμου υπάρχουν πολλαπλές γραμμές που έχουν ανιχνευθεί. Οπότε αφού ομαδοποιήσουμε τις γραμμές (με κριτήριο την κλίση τους και την μεταξύ τους απόσταση, μέσα από την απόσταση των σημείων των ευθύγραμμων τμημάτων) κρατάμε τις ομάδες εκείνες που αποτελούνται από τουλάχιστον 2 γραμμές. Τέλος για κάθε μία από τις ομάδες αυτές δημιουργούμε μία αντιπροσωπευτική γραμμή, που θα έχει κλίση τον μ.ο. των κλίσεων (αφού όλες θα είναι παρόμοιες όπως αναφέρθηκε παραπάνω) και μήκος το μεγαλύτερο δυνατόν.



Με πράσινο φαίνονται οι γραμμές που τελικά θα προκύψουν (για 2 διαφορετικές εικόνες εισόδου).

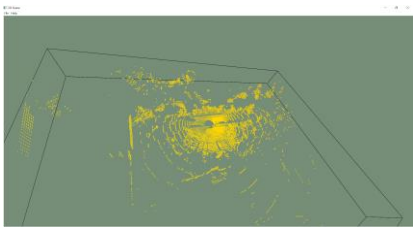
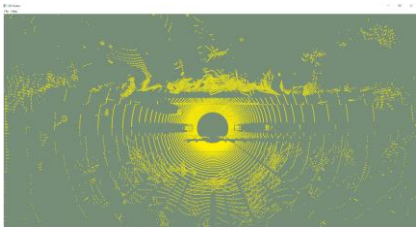
b. Kitti

Τα αποτελέσματα (κρίνοντας από το προ-τελευταίο στάδιο ανίχνευσης ακμών που έχει υλοποιηθεί) είναι πολύ διαφορετικά. Ενώ όπως φαίνεται από τις εικόνες Carla οι γραμμές του δρόμου θα ανιχνεύονταν με επιτυχία (αφού για τις κύριες γραμμές του δρόμου στην εικόνα ανιχνεύονται πάνω από 2 ακμές κάθε φορά) στις εικόνες Kitti δεν θα είχαμε αποτελεσματική ανίχνευση των γραμμών. Εκτός από την άρση του περιορισμού που αναφέρθηκε παραπάνω (ότι κρατάμε τις ομάδες γραμμών που έχουν τουλάχιστον 2 γραμμές) χρειάζονται και άλλες αλλαγές για να καταστεί αποτελεσματικό το παραπάνω πρόγραμμα για ανίχνευση ακμών στο περιβάλλον του Kitti.

Όπως φαίνεται έχει ανιχνευτεί μόνο μία γραμμή.

Εδώ έχει ανιχνεύσει περισσότερες γραμμές αλλά λανθασμένες.

2. Lidar-space

a. Εκτίμηση του εδάφους (Carla).

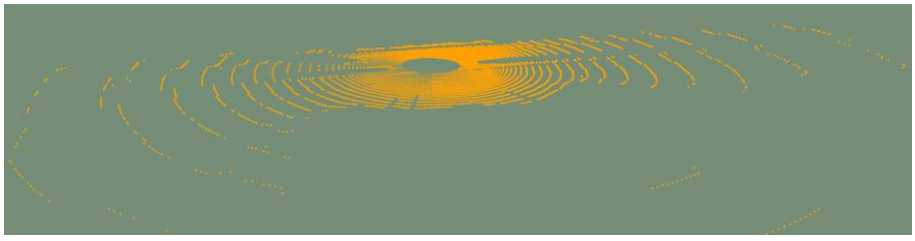
i. Threshold στην z συνιστώσα.

Από το αρχικό point cloud (point_cloud_original) θεωρούμε έδαφος το σημεία τα οποία έχουν συντεταγμένη z στο πρώτο 2% του συνολικού εύρους της. Οπότε αντίστοιχα αποθηκεύουμε τους δείκτες των σημείων αυτών σε έναν πίνακα.

ii. Ransac.

Θεωρούμε ότι το έδαφος προσομοιάζει σε ένα επίπεδο, για αυτό χρησιμοποιούμε τον αλγόριθμο Ransac.
Ο αλγόριθμος Ransac (plane) ακολουθεί την εξής λογική (με βάση την υλοποίησή του στον κώδικα):
-Επιλέγουμε τυχαία 3 σημεία από το point cloud και υπολογίζουμε τις παραμέτρους του (γεωμετρικού) επιπέδου που περνάει από αυτά τα 3 σημεία.
-Έπειτα για κάθε ένα από τα υπόλοιπα σημεία ελέγχουμε αν αυτά ταιριάζουν με το συγκεκριμένο μοντέλο του επιπέδου (δηλαδή αν απέχουν απόσταση μικρότερη από το δοσμένο όριο).
-Επαναλαμβάνουμε την διαδικασία αυτή για ένα δοσμένο αριθμό επαναλήψεων και κρατάμε το καλύτερο μοντέλο (δηλαδή το μοντέλο το οποία έχει τελικά τα περισσότερα ταιριαστά στοιχεία).

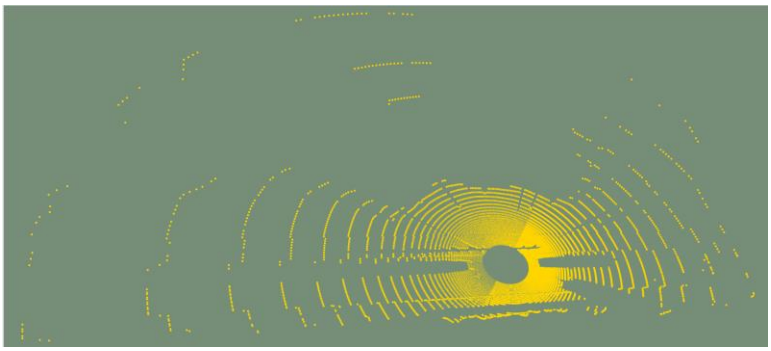
Έτσι έχουμε υπολογίσει-εκτιμήσει τα σημεία του εδάφους και όμοια με πριν αποθηκεύουμε τους δείκτες των σημείων σε έναν πίνακα.



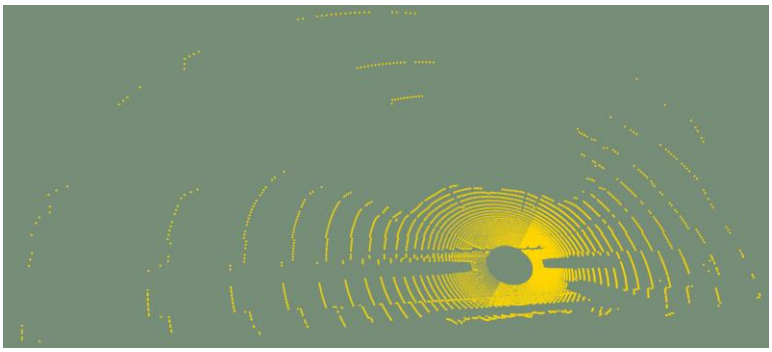
Η διαφορά στις 2 υλοποιήσεις είναι η εξής:

- Η πρώτη μέθοδος έχει καλύτερα αποτελέσματα όταν το έδαφος είναι παράλληλο στον άξονα z. Αν για παράδειγμα το έδαφος έχει κάποια κλίση (σε σχέση με τον z άξονα) τότε δεν θα το εκτιμήσει σωστά. Επιπλέον δεν επηρεάζεται η αποτελεσματικότητά του από τυχόν λακκούβες στο έδαφος και από επιφάνειες μεγαλύτερες του εδάφους.
- Η δεύτερη μέθοδος είναι ανεξάρτητη από την κλίση του εδάφους. Όμως δεν είναι τόσο αποτελεσματική αν το έδαφος παρουσιάζει σχετικά έντονες ανωμαλίες (δηλαδή αν δεν προσομοιάζει σε ένα επίπεδο, π.χ. έχει βαθιές λακκούβες ή μεγάλα εξογκώματα) ή αν το point cloud περιέχει κάποιον κατακόρυφο τοίχο (ή γενικότερα επιφάνεια διαφορετική του εδάφους) πολύ μεγάλο σε σχέση με το έδαφος που αποτυπώνεται.

Παρόλα αυτά δεδομένου ότι τα «αδύναμα» σημεία των μεθόδων δεν υπάρχουν σε μεγάλο βαθμό στην συγκεκριμένη περίπτωση οι δύο μέθοδοι εκτιμούν με πολύ μεγάλη ομοιότητα το έδαφος.

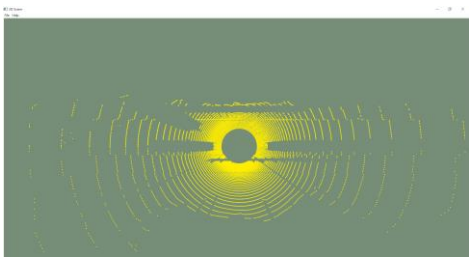


1η μέθοδος

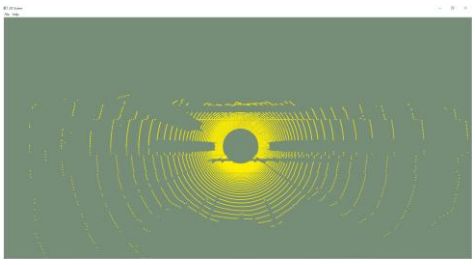


2η μέθοδος

1η



2η



Για τα επόμενα ερωτήματα χρησιμοποιείται ο πίνακας με τα σημεία εκτός του εδάφους που υπολογίστηκε με την πρώτη μέθοδο (z threshold). Ο λόγος είναι απλώς και μόνο επειδή είναι ευκολότερη η δημιουργία ενός νέου point cloud με σημεία τα σημεία του αρχικού point cloud που δεν εκτιμήθηκαν ως έδαφος.

b. Ομαδοποίηση και σύγκριση αντικειμένων.

Για την απομόνωση του εδάφους λαμβάνουμε υπόψιν όλα τα σημεία εκτός του point cloud εκτός από αυτά των οποίων οι δείκτες εμπεριέχονται στον παραπάνω πίνακα (δηλαδή τον πίνακα των σημείων εδάφους).

Για να συγκριθούν τα αντικείμενα που βρίσκονται πάνω από το έδαφος με τα δοσμένα πρότυπα πρέπει πρώτα να ομαδοποιηθούν τα σημεία του point cloud (που περιέχει τα σημεία εκτός του εδάφους) ώστε να μπορέσουν να αντιμετωπιστούν σαν ξεχωριστά αντικείμενα.

Για την ομαδοποίηση χρησιμοποιείται ο αλγόριθμός DBSCAN. Στην συγκεκριμένη περίπτωση είναι αρκετά αποτελεσματικός σε σχέση με άλλους γνωστούς αλγόριθμους ομαδοποίησης καθώς πρώτον δεν γνωρίζουμε από την αρχή το πλήθος των αντικειμένων και δεύτερον δεν επηρεάζεται από το σχήμα των αντικειμένων-ομάδων. Για παράδειγμα οι δύο προηγούμενες συνθήκες καθιστούν όχι και τόσο αποδοτικό τον αλγόριθμο K-means. Επιπλέον ο DBSCAN είναι ανθεκτικός στον θόρυβο/ στα outliers σημεία.

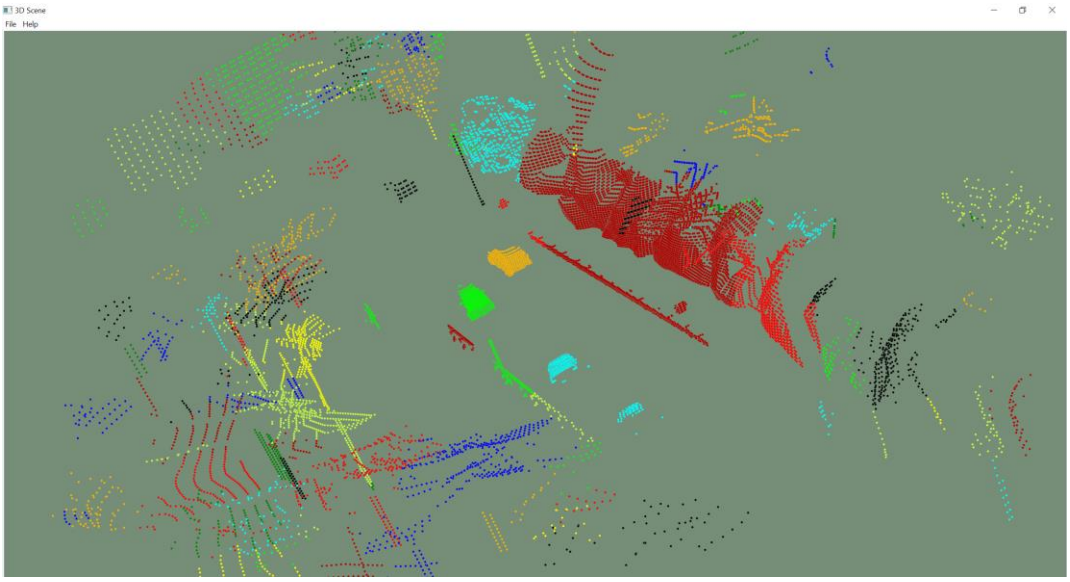
Γενικά ο αλγόριθμος DBSCAN ακολουθεί την εξής λογική:

(-Γείτονες σημείου θεωρούνται τα σημεία που απέχουν μικρότερη απόσταση από ένα δοσμένο όριο.
-Κάθε σημείο κατατάσσεται σε από τις 3 κατηγορίες 1) Θόρυβος 2) Σημεία πυρήνα (core points) και 3) Γείτονες σημείων πυρήνα. Στην $2^η$ κατηγορία ανήκουν τα σημεία που έχουν γείτονες σε πλήθος μεγαλύτερο από το δοσμένο όριο. Στην $3^η$ κατηγορία ανήκουν οι γείτονες των σημείων πυρήνα που δεν είναι σημεία πυρήνα.)

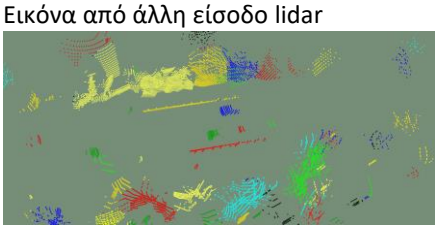
-Αρχικά βρίσκουμε τους γείτονες κάθε σημείου και αναγνωρίζουμε τα σημεία πυρήνα.
-Βρίσκουμε τα core points που είναι μεταξύ τους γείτονες και δημιουργούμε τις αντίστοιχες ομάδες.
-Κατατάσσουμε κάθε ένα σημείο που δεν είναι core point σε μία ομάδα που γειτνιάζει, αν δεν γειτνιάζει σε καμία τότε σε θόρυβο.

Αναλυτικότερα ο αλγόριθμός στην συγκεκριμένη περίπτωση ακολουθεί την λογική που εξηγείται καλύτερα στον σύνδεσμο.

Για την εύρεση γειτόνων χρησιμοποιείται μία βοηθητική συνάρτηση η οποία ελέγχει όλα τα σημεία του point cloud (χωρίς το έδαφος) αν απέχουν απόσταση (από το δοσμένο σημείο) μικρότερη της δοσμένης τιμής.



Εικόνα από άλλη είσοδο lidar



Αφού ξεχωρίσαμε τα αντικείμενα μπορούμε πλέον να τα συγκρίνουμε.

Για πρότυπα αντικειμένων χρησιμοποιούνται τα 4 δοσμένα (2 μοντέλα αυτοκινήτων και 2 πεζών)
Για την σύγκριση χρησιμοποιούνται οι εξής μετρικές:

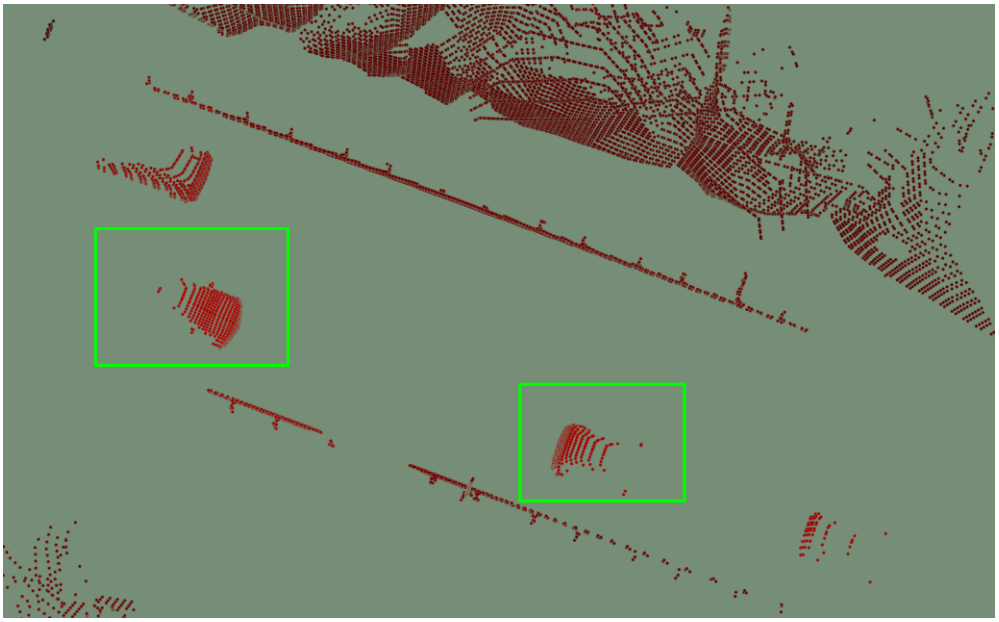
1. Η διαφορά των κατώτατων σημείων στον z άξονα
2. Η διαφορά του εύρους των τιμών των συντεταγμένων στον z άξονα
3. *(Αφού μετασχηματίσουμε τα σημεία σε κυλινδρικές συντεταγμένες r,θ,z)*
    a. Την διαφορά των διασπορών της μεταβλητής r
    b. Την διαφορά των μέγιστων τιμών της μεταβλητής r

Με συντελεστές 0.3 , 0.03 , 0.2 , 0.2 αντίστοιχα για τον υπολογισμό της συνολικής μεταβλητής που εκφράζει την ομοιότητα των 2 αντικειμένων.

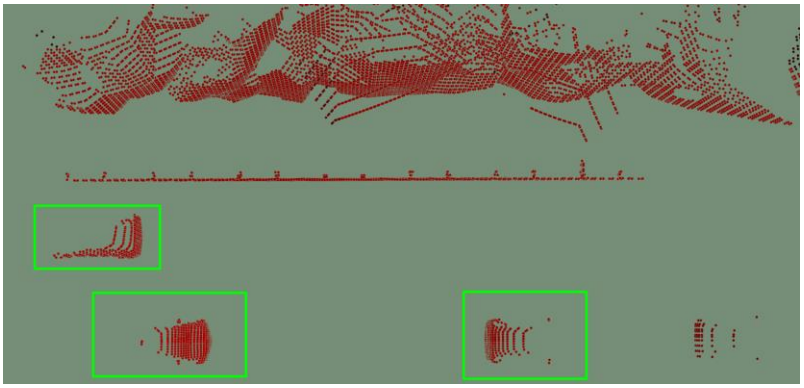
Η μεταβλητή ομοιότητας παίρνει τιμές από 0 (διαφορετικά αντικείμενα) έως 1 (παρόμοια-ίδια αντικείμενα).

Χρησιμοποιήθηκαν οι παραπάνω μετρικές λόγω των συνθηκών της σκηνής-point cloud (συμπεριλαμβανομένου και του περιεχομένου του μοντέλου σύγκρισης). Δηλαδή σε μία σκηνή δρόμου όπως αυτές (με επίπεδο σχετικά έδαφος) ένα αυτοκίνητο θα βρίσκεται κοντά στο έδαφος ($1^η$ μετρική), επιπλέον θα έχει παρόμοιο ύψος ($2^η$ μετρική) με το μοντέλο, και τέλος θα έχει παρόμοιο σχήμα στους x,y άξονες (όχι ξεχωριστά σε κάθε έναν αλλά και στους 2 μαζί γιατί μπορεί να έχει περιστρέφει στον z άξονα, για αυτό χρησιμοποιούνται οι κυλινδρικές συντεταγμένες) ($3^η$ μετρική).

Τέλος για να αποτυπώσουμε την τιμή ομοιότητας (σε σχέση με το πρώτο πρότυπο αμαξιού για παράδειγμα) εμφανίζουμε τα αντικείμενα με χρώμα (στην κόκκινη μεταβλητή χρώματος) ανάλογο της ομοιότητας, δηλαδή με έντονο κόκκινο τα αντικείμενα που μοιάζουν πολύ με το μοντέλο του αυτοκινήτου και αχνά κόκκινα αυτά που δεν μοιάζουν. Θα μπορούσαμε επιπλέον να κάνουμε πιο έντονη την διαφορά ομοιότητας μεταξύ 2 αντικειμένων με το πρότυπο εφαρμόζοντας έναν γραμμικό μετασχηματισμό στην μεταβλητή ομοιότητας.

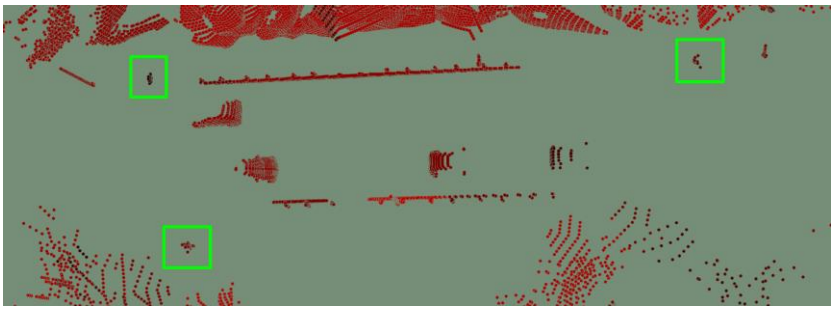


Δεν είναι έντονη η διαφορά άλλα φαίνεται ότι τα 2 αντικείμενα μοιάζουν περισσότερο από τα υπόλοιπα(εντονότερο χρώμα).

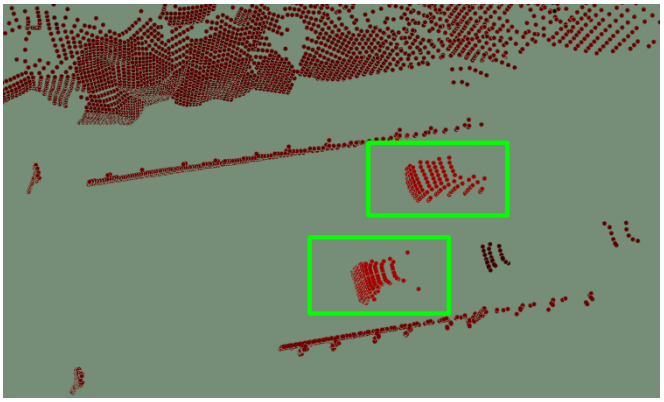


Στο 2ο μοντέλο αυτοκινήτου είναι πάλι εμφανής η διαφορά.

Στην περίπτωση της σύγκρισης με το μοντέλο πεζού τα αποτελέσματα δεν είναι και τόσο καλά εξαιτίας της μεγαλύτερης λεπτομέρειας που χρειάζεται για την σύγκριση και του μικρού μεγέθους του μοντέλου.



Εικόνα από άλλη είσοδο lidar



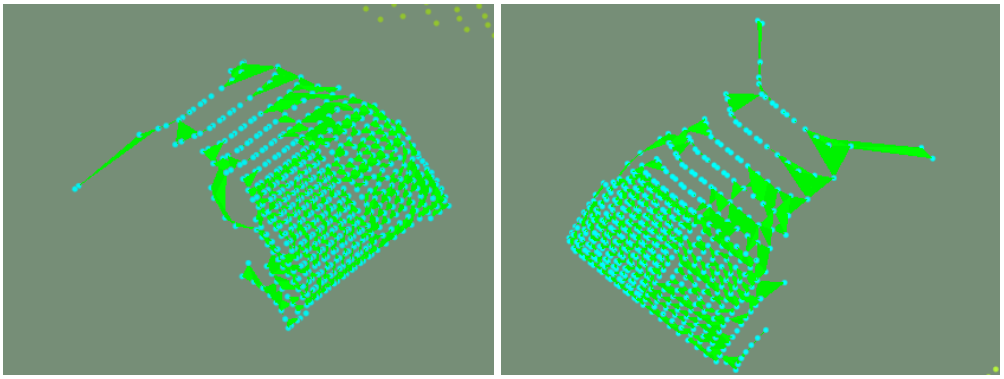
c. Τριγωνοποίηση.

Σημείωση : Για την τριγωνοποίηση των αντικειμένων θεωρούμε ότι ζητείται όλων των αντικειμένων (εκτός τους εδάφους) και όχι μόνο των αντικειμένων που ακουμπούν στο έδαφος (επειδή μπορεί να παρερμηνευτεί με βάση την εκφώνηση). Σε αντίθετη περίπτωση απλώς θα ελέγχαμε αν το κατώτερο σημείο του κάθε αντικειμένου (ως προς τον άξονα z) είναι κοντά στην τιμή της συντεταγμένης z του εδάφους.

Για την τριγωνοποίηση των αντικειμένων χρησιμοποιείται ένας αλγόριθμος ο οποίος για κάθε σημείο βρίσκει τα 4 γειτονικότερά του σημεία και προσθέτει στον πίνακα των τριγώνων τα 4 τρίγωνα που προκύπτουν από τα 5 συνολικά σημεία (με την λογική ότι τα σημεία, που αποτελούν το περίγραμμα του αντικειμένου και άρα μπορούν να αντιμετωπιστούν σαν στις 2 διαστάσεις για καλύτερη κατανόηση, σχηματίζουν περίπου ένα σταυρό με κέντρο το σημείο ενδιαφέροντος).
Για να υπολογιστούν οι παραπάνω γείτονες χρησιμοποιείται μία βοηθητική συνάρτηση που ταξινομεί σε έναν πίνακα όλα τα σημεία εκτός του σημείου που ελέγχουμε με βάση την απόσταση τους από αυτό με χρήση των τεχνικών δυαδικής αναζήτησης και προσθήκης τιμής σε ταξινομημένο πίνακα. Τέλος επιλέγονται οι 4 πρώτοι (και άρα πλησιέστεροι) γείτονες. Υπάρχει η δυνατότητα ταχύτερης υλοποίησης του αλγορίθμου αν βάλουμε ένα όριο στην απόσταση που μπορεί να έχουν οι γείτονες και έτσι να μειωθεί το μέγεθος του πίνακα. Αυτό όμως δημιουργεί προβλήματα στα ακραία σημεία. Επιπλέον υπάρχει η δυνατότητα βελτίωσης του αλγορίθμου εύρεσης γειτόνων αν κρατάει κάθε φορά τα 4 κοντινότερα σημεία.

Επειδή με αυτήν την λογική δημιουργούνται πολλαπλά στιγμιότυπα του ίδιου τριγώνου υλοποιούμε μία συνάρτηση η οποία διαγράφει τα διπλότυπα τρίγωνα.

Επειδή ο αλγόριθμος δεν είναι πολύ αποδοτικός (αλλά και χρονοβόρος) η τριγωνοποίηση γίνεται και εμφανίζεται σε ένα αντικείμενο μόνο για εξοικονόμηση χρόνου λειτουργίας του προγράμματος.



Πηγές:

- https://docs.opencv.org/
- https://www.youtube.com/watch?v=BgA9asLpkJU
- https://www.analyticsvidhya.com/blog/2020/05/tutorial-real-time-lane-detection-opencv/
- https://learnopencv.com/hough-transform-with-opencv-c-python/
- https://stackoverflow.com/questions/49771659/how-to-filter-hough-lines-for-multi-lane-detection

- https://en.wikipedia.org/wiki/Random_sample_consensus
- https://medium.com/@ajithraj_gangadharan/3d-ransac-algorithm-for-lidar-pcd-segmentation-315d2a51351
- https://en.wikipedia.org/wiki/DBSCAN#Abstract_algorithm
- https://towardsdatascience.com/how-to-automate-3d-point-cloud-segmentation-and-clustering-with-python-343c9039e4f5
- https://medium.com/@ajithraj_gangadharan/euclidean-clustering-for-lidar-point-cloud-data-8603f266b246