ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ - ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

"Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας" Ακαδημαϊκό Έτος 2020-21 (Εαρινό Εξάμηνο)

Εργαστηριακές Ασκήσεις - Μέρος Β΄

Μαναγούδης Μιχαήλ 1059398

Περιγραφή

Η εργασία υλοποιήθηκε σε περιβάλλον Matlab. Επιλέχθηκε το 4ο θέμα προς λύση. Αντικείμενο αυτού του θέματος είναι η επεξεργασία των καρέ ενός βίντεο από κάμερα οχήματος με σκοπό την εξαγωγή μίας περιοχής ενδιαφέροντος και την ανίχνευση κινούμενων οχημάτων. Επιπλέον ζητείται ο σχολιασμός και η παρατήρηση της ανοχής των παραπάνω αλγορίθμων σε συνθήκες θορύβου.

Στο αρχείο κώδικα υπάρχουν και τα κατάλληλα σχόλια που επεξηγούν την διαδικασία υλοποίησης των ζητουμένων του επιλεγμένου θέματος .

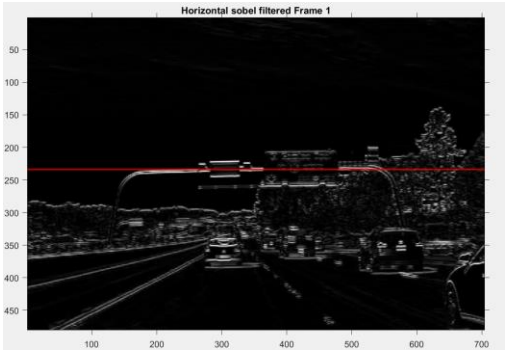
1. Δημιουργίας περιοχής ενδιαφέροντος

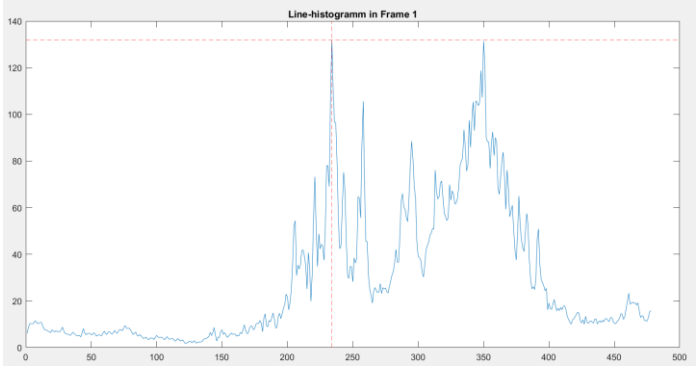
Αρχικά φορτώνουμε το βίντεο και αποθηκεύουμε τα καρέ (frames) σε έναν πίνακα. Από τα χαρακτηριστικά του βλέπουμε ότι αποτελείται από 300 εικόνες (frames) διάστασης 480x704 και επιπλέον το βίντεο είναι 30fps.

Frame 1

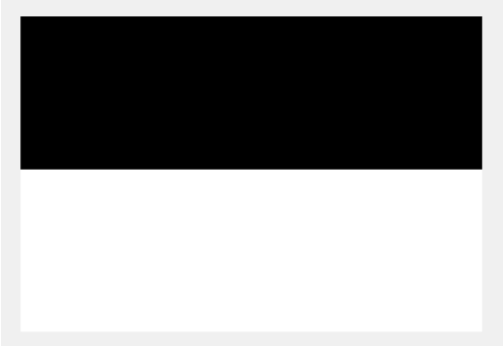
Η παρακάτω επεξήγηση του 1ου αλγορίθμου αναφέρεται στην εφαρμογή του σε ένα frame του βίντεο(στον κώδικα υλοποιείται φυσικά σε ολόκληρο το βίντεο). Για την υλοποίηση του αλγορίθμου λαμβάνουμε υπόψιν μας το αντικείμενο και την γωνία που αυτό απεικονίζεται . Έτσι ενδιαφερόμεστε κυρίως για τα οριζόντια χαρακτηριστικά της εικόνα. Για αυτό λοιπόν αρχικά εφαρμόζουμε ένα οριζόντιο παράθυρο Sobel. Η γραμμή με το μέγιστο άθροισμα είναι η γραμμή απο την οποία ξεκινάει η περιοχή ενδιαφέροντος ROI και καταλήγει μέχρι το κάτω μέρος της εικόνας.Οι υπόλοιπες γραμμές θεωρούνται εκτός του ROI. Για την εφαρμογή του φίλτρου Sobel μετατρέπουμε την εικόνα σε ασπρόμαυρη και έπειτα κάνουμε συνελιξη με την οριζόντια μήτρα Sobel. Στη συνέχεια κανονικοποιούμε στο διάστημα [-1,1]. Η τιμή του κάθε pixel μας δείχνει την πληροφορία για την ακμή και το πρόσημο απλώς την κατεύθυνσή της. Για αυτό κρατάμε την απόλυτη τιμή του.

Έπειτα κατασκευάζουμε ένα “ιστόγραμμα” για να αποτυπώσουμε την οριζόντια πληροφορία. Το ιστόγραμμα αυτό αποτυπώνει το άθροισμα των απολύτων τιμών ανα γραμμή της εικόνας. Παρακάτω με κόκκινη γραμμή φαίνεται η γραμμή με το μέγιστο άθροισμα αυτό.





Αφού βρούμε την γραμμή αυτή κατασκευάζουμε την μάσκα. Η μάσκα είναι ένας πίνακας διαστάσεων όσο της εικόνας όπου τα στοιχεία πάνω από την γραμμή που αναφέρθηκε είναι 0 και τα από κάτω 1. Τέλος εφαρμόζουμε την μάσκα στην εικόνα, πολλαπλασιάζοντας κάθε pixel της εικόνας με την αντίστοιχη τιμή της μάσκας(για κάθε χρωματικό κανάλι).



Μάσκα Frame

Εικόνα μετά την εφαρμογή της μάσκας.

Υλοποιείται συνάρτηση που αποθηκεύει τα frames μετά την εφαρμογή μάσκας σε αρχείο βίντεο.

Τα αποτελέσματα αυτής της μεθόδου, αν δούμε και τα υπόλοιπα frames, δεν είναι τόσο ικανοποιητικά.

Η μάσκα αλλάζει αρκετά από το ένα frame στο άλλο και συχνά καλύπτει μέρος (έως κα τελείως όπως φαίνεται παραπάνω) τον οχημάτων. Για να έχουμε καλύτερα αποτελέσματα υλοποιείται ο 2ος αλγόριθμος που αποτελεί βελτίωση του 1ου. Συγκεκριμένα αυτήν την φορά παίρνουμε σαν γραμμή που οριοθετεί την μάσκα όχι την γραμμή με το μέγιστο άθροισμα αλλά την πρώτη που συναντάμε με άθροισμα ένα ποσοστό του μέγιστου (έπειτα από πειραματισμό 80%) αφού πρώτα κάνουμε ένα smoothing στο ιστόγραμμα (mean filter 3x3) για να αποφύγουμε ακραίες τιμές που δεν μας αφορούν. Για να γίνει επιπλέον ταχύτερος ο αλγόριθμος χρησιμοποιούμε την γνώση από τα προηγούμενα frames. Αναζητούμε αυτήν την γραμμή στο διάστημα 10 γραμμών πριν έως 10 μετά από την γραμμή του προηγούμενου frame. Αν δεν είναι το άθροισμά της μεγαλύτερο απο το ποσοστό της μέγιστης που θέλουμε τότε αναζητούμε σε όλη την εικόνα. Επίσης για να αποφύγουμε περιπτώσεις που η γραμμή-όριο αλλάξει κατα πολύ ανα 10 frame

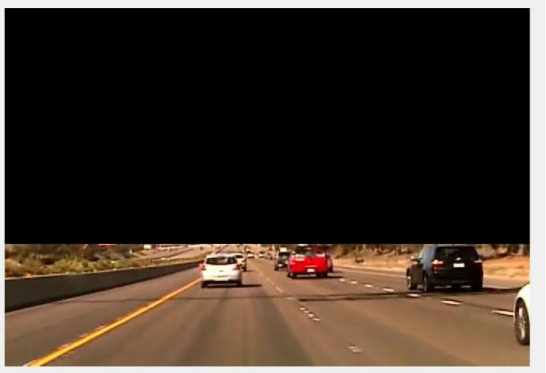
ελέγχουμε ολόκληρη την εικόνα.

Αλγόριθμος 2ος (βελτιωμένος)

Για τα ίδια frame:

1ος Αλγόριθμος

2ος Αλγόριθμος

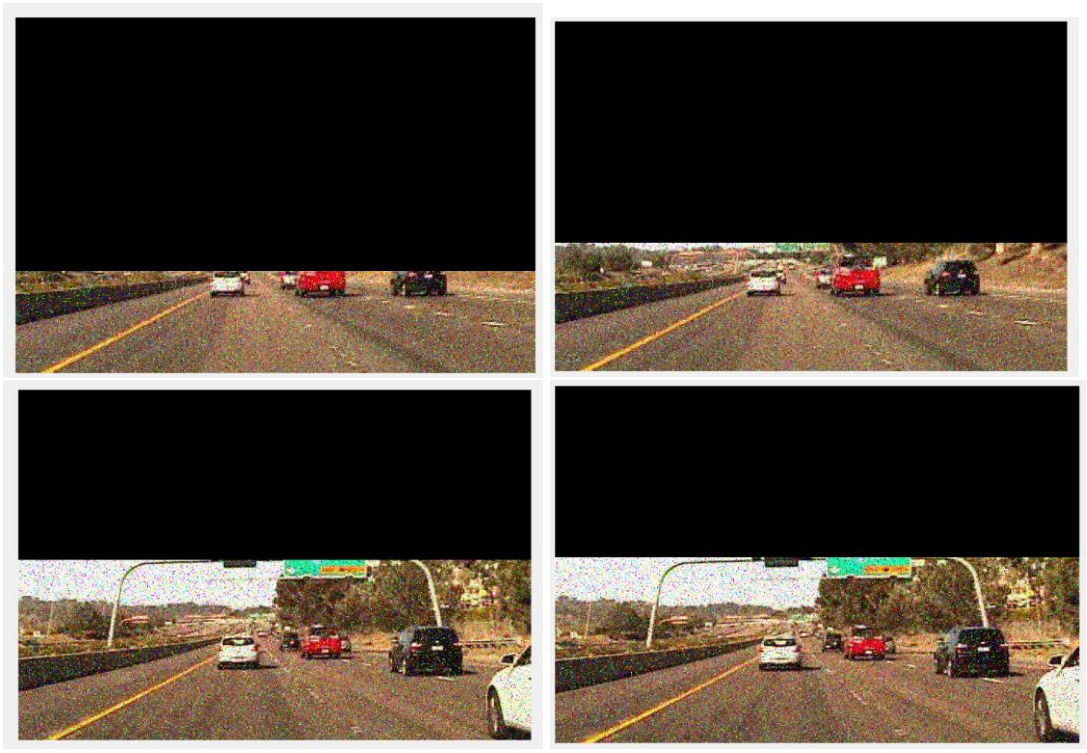


Παρατηρούμε (στα περισσότερα frames) ότι όντως η μάσκα βελτιώθηκε κατα πολύ καθώς κόβει λιγότερη περιοχή από τα οχήματα. Επίσης αν δούμε τον χρόνο εκτέλεσης των 2 αλγορίθμων ο 2ος σχεδόν όλες τις φορές είναι γρηγορότερος (είναι μικρή η διάρκεια και των 2 σε μεγαλύτερους χρόνους φαίνεται ακόμα καλύτερα η διαφορά). Παρόλα αυτά οι οριζόντιες γραμμές (για παράδειγμα της πινακίδας πάνω απο τον δρόμο) μπερδεύουν τον αλγόριθμο και αναγνωρίζει μεγαλύτερη περιοχή ενδιαφέροντος από την ζητούμενη, αλλά αυτό είναι καλύτερο από το να κόβει απο την περιοχή χρήσιμα στοιχεία των οχημάτων.

2. Θόρυβος

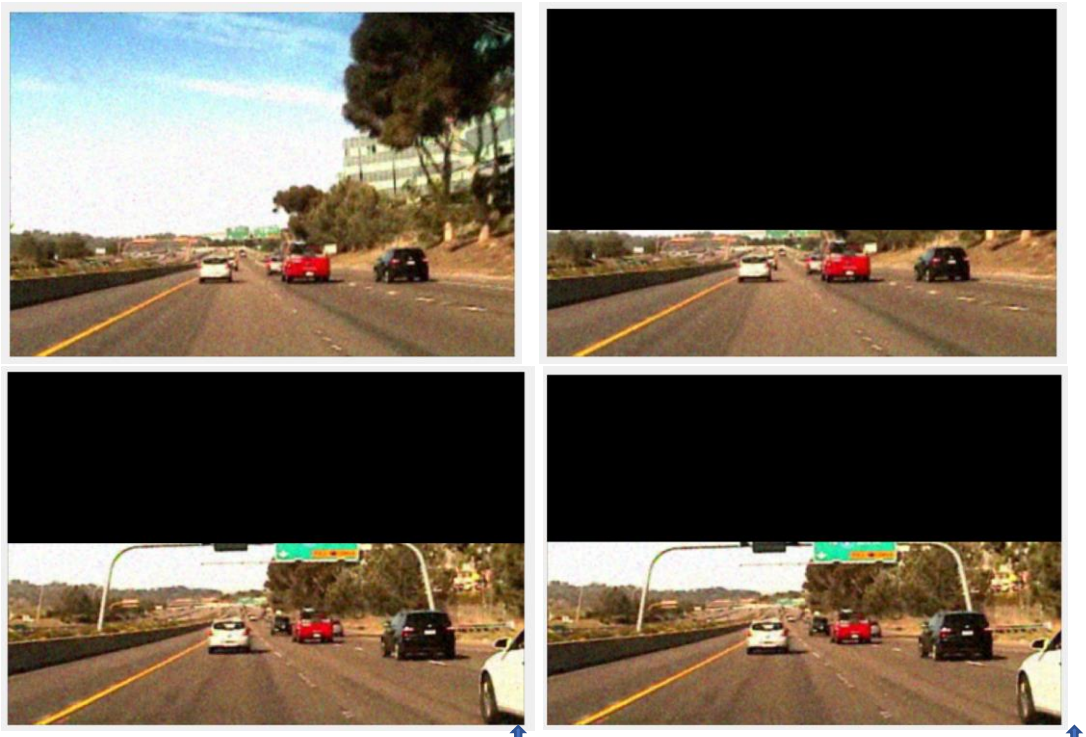
Υποβαθμίζουμε το βίντεο με θόρυβο προσθέτοντας σε κάθε frame λευκό gaussian θόρυβο και κρουστικό. Η διαδικασία είναι όμοια με αυτήν που αναπτύχθηκε στο 1ο μέρος των ασκήσεων. Όμοια με το 1ο μέρος των ασκήσεων είναι επίσης η διαδικασία αφαίρεσης του θορύβου (πρώτα με ένα ένα median φίλτρο που θα αφαιρέσει τον κρουστικό θόρυβο και έπειτα με ένα average φίλτρο που θα αφαιρέσει μέρους του gaussian). Επειδή η διαδικασία της αποθορυβοποίησης (η οποία σημειοτέον πρέπει να εφαρμοστεί και στα 3 χρωματικά κανάλια και για τις 300 εικονες) παίρνει πολύ χρόνο αφού έτρεξε μία φορά για αρκετή ώρα ο αλγόριθμος αποθήκευσαμε το αποτέλεσμα σε αρχείο (γραμμή 74-75) ώστε κάθε φορά να διαβάζουμε το αρχείο σε πιο σύντομο χρονικό διάστημα σε σχέση με το να εφαρμόζαμε κάθε φορά τον αλγόριθμο αποθορυβοποίησης.

Αποτελέσματα:



Αλγόριθμος 1ος με θόρυβο

Αλγόριθμος 2ος με θόρυβο



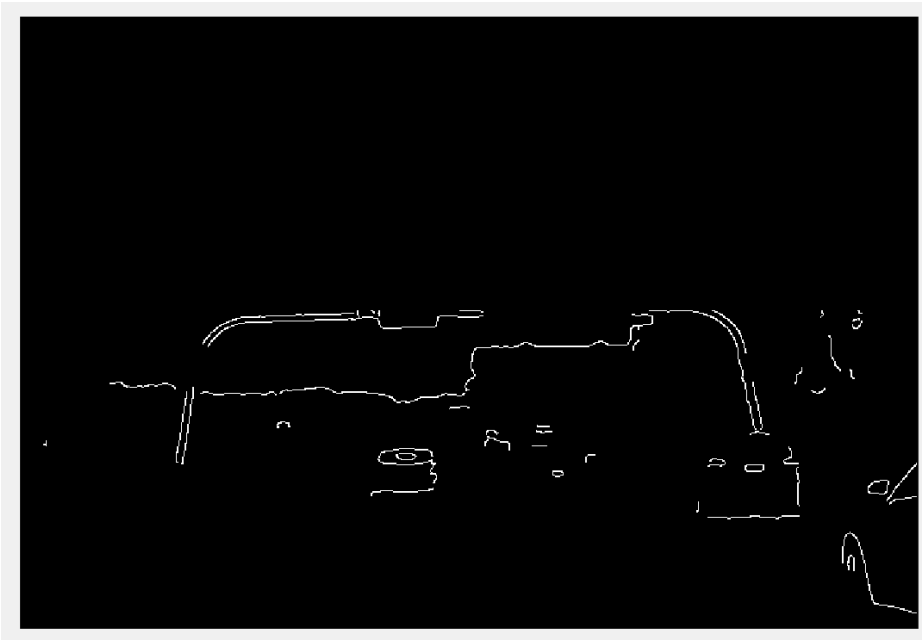
Αλγόριθμος 1ος μετά από αποθορυβοποίηση

Αλγόριθμος 2ος μετά από αποθορυβοποίηση

Η μέθοδος εξακολουθεί να δουλεύει αποτελεσματικά και στην περίπτωση του θορύβου αλλά ακόμα περισσότερο στην περίπτωση της αποθορυβοποίησης. Απλώς κάποιες φορές είναι πιο ευαίσθητη και επιλέγει μεγαλύτερη περιοχή ενδιαφέροντος από την ζητούμενη. Ένας τρόπος αντιμετώπισής της είναι να μεγαλώσουμετο κατώφλι του ποσοστό της μέγιστης τιμής του ιστογράμματος. Ο θόρυβος μειώνεται σημαντικά κατά την διαδικασία της αποθορυβοποίησης. Τελικά φαίνεται ότι η παρουσία θορύβου δεν επηρέασε σημαντικά τον αλγόριθμο.

3. Ανίχνευση ακμών

Για την ανίχνευση του οχήματος θα χρησιμοποιήσουμε την μέθοδο ανίχνευσης ακμών Canny. Ο αλγόριθμος υλοποιείται μέσω της αντίστοιχης εντολής matlab αφού μετατρέψουμε την κάθε εικόνα σε ασπρόμαυρη. Η διαφορά είναι ότι τον εφαρμόζουμε μόνο στην περιοχή ενδιαφέροντος και όχι σε όλη την εικόνα. Αυτο επιτυγχάνεται με τους βοηθητικούς πίνακες lines… που αποθηκεύουν για κάθε frame τον αριθμό της οριζόντιας γραμμής που ξεκινάει η ROI. Έπειτα απο πειραματισμούς καταλήξαμε στα κατώφλια 0.4 και 0.6 για τον αλγόριθμο Canny.



Σημείωση:
Το video που περιέχει τις εικόνες που
αποτελούνται μόνο από την περιοχή ενδιαφέροντος είναι το “video_masked_2.avi”.
Αντίστοιχα αποθηκεύονται σε αρχεία βίντεο τα αποτελέσματα των διαφόρων
σταδίων του κώδικα.