ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΙΕΣ

ΨΗΦΙΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΒΙΝΤΕΟ

ΑΣΚΗΣΗ 2

Ονοματεπώνυμο:	Βασιλική Στάμου
Αριθμός Μητρώου:	1059543
Τομέας:	Μεταπτυχιακό ΣΜΗΝ
Έτος φοίτησης:	1ο
Ακ. έτος διεξαγωγής μαθήματος:	2023-2024
Ημ/νία διεξαγωγής Φ.Α.:	03.04.2024

Περιεχόμενα

1 Υλοποιήστε έναν αλγόριθμο tracking χρησιμοποιώντας φίλτρο σωματιδίων για το βίντεο sevenup.m4v. 1

2 Προσπαθήστε να επιλύσετε το πρόβλημα επεκτείνοντας το μοντέλο σας ώστε να συμπεριλάβετε τον προσανατολισμό, που θα αναπαριστά με τη περιστροφή του κουτιού και χρησιμοποιήστε το στην ακολουθία coke. 8

3 (Προαιρετικό) Υλοποιήστε τα ερωτήματα 1,2 με χρήση φίλτρου Kalman. 10

3.1 Θεωρία του Φίλτρου Kalman 10

3.2 Φίλτρο Kalman σε μια διάσταση 11

4 Κώδικας 14

4.1 Αλγόριθμος tracking με φίλτρο σωματιδίων 14

4.2 Αλγόριθμος tracking με φίλτρο σωματιδίων & προσανατολισμός 17

4.3 Αλγόριθμος tracking με φίλτρο Kalman 21

1 Υλοποιήστε έναν αλγόριθμο tracking χρησιμοποιώντας φίλτρο σωματιδίων για το βίντεο sevenup.m4v.

Η παρακολούθηση αντικειμένων είναι ένα κρίσιμο στοιχείο στην υπολογιστική όραση, το οποίο περιλαμβάνει τον εντοπισμό και την παρακολούθηση της κίνησης ενός αντικειμένου μέσω μιας ακολουθίας πλαισίων βίντεο. Μια αποτελεσματική μέθοδος παρακολούθησης είναι η χρήση Φίλτρου Σωματιδίων, το οποίο είναι ιδιαίτερα χρήσιμο στην αντιμετώπιση μη γραμμικών και μη-Γκαουσιανών μοντέλων. Αυτή η αναφορά παρουσιάζει την υλοποίηση ενός συστήματος παρακολούθησης βασισμένο σε φίλτρο σωματιδίων για ένα βίντεο (sevenup.m4v), όπου το αντικείμενο-στόχος μοντελοποιείται ως ορθογώνιο με σταθερό μέγεθος και η θέση του εκτιμάται μέσω παρατηρήσεων βασισμένων σε ακμές.

Ορισμός του Προβλήματος

Ο στόχος είναι η υλοποίηση ενός φίλτρου σωματιδίων για την παρακολούθηση ενός ορθογώνιου αντικειμένου σε ένα βίντεο. Η θέση του αντικειμένου ορίζεται από τις συντεταγμένες του κέντρου του (x, y), ενώ οι διαστάσεις του ορθογωνίου παραμένουν σταθερές καθ' όλη τη διάρκεια του βίντεο. Το μοντέλο παρατήρησης για αυτή την παρακολούθηση βασίζεται στον εντοπισμό ακμών, συγκεκριμένα σε χάρτες ακμών που δημιουργούνται μέσα σε ένα παράθυρο γύρω από την προβλεπόμενη θέση του αντικειμένου.

Η πιθανοφάνεια της παρουσίας του αντικειμένου σε μια δεδομένη θέση υπολογίζεται χρησιμοποιώντας τον μετασχηματισμό απόστασης των εντοπισμένων ακμών, εφαρμόζοντας μια συνάρτηση που εξασθενεί εκθετικά με την απόσταση των εντοπισμένων ακμών από τα όρια του ορθογωνίου.

Μαθηματικό Μοντέλο και Επισκόπηση του Φίλτρου Σωματιδίων

Μοντέλα Κατάστασης και Παρατήρησης

Μοντέλο Κατάστασης: Η κατάσταση του συστήματος είναι η θέση του κέντρου του ορθογωνίου (x, y). Η δυναμική του συστήματος μοντελοποιείται με μια διαδικασία θορύβου, όπου η κατάσταση εξελίσσεται με βάση μια κανονική κατανομή με κέντρο την τρέχουσα κατάσταση και συνδιακύμανση V**.**

Μοντέλο Παρατήρησης: Η παρατήρηση βασίζεται στις ακμές που εντοπίζονται μέσα σε ένα παράθυρο γύρω από την προβλεπόμενη θέση του αντικειμένου. Η συνάρτηση πιθανοφάνειας ορίζεται ως:

$$p(x|z)= e^{-a\cdot d}$$

όπου a είναι ένας συντελεστής εξασθένησης και d είναι η μέση απόσταση των ακμών από τα σημεία δειγματοληψίας στις πλευρές του ορθογωνίου. Η απόσταση υπολογίζεται χρησιμοποιώντας τον μετασχηματισμό απόστασης bwdist() στην εικόνα των εντοπισμένων ακμών.

Αλγόριθμος Φίλτρου Σωματιδίων

Το Φίλτρο Σωματιδίων (γνωστό και ως Μέθοδος Monte Carlo Σειριακής Μορφής) λειτουργεί ως εξής:

Αρχικοποίηση: Τα σωματίδια αρχικοποιούνται γύρω από την αρχική εκτιμώμενη θέση του αντικειμένου. Τα σωματίδια κατανέμονται σύμφωνα με μια κανονική κατανομή με καθορισμένη συνδιακύμανση V.

Πρόβλεψη: Η θέση κάθε σωματιδίου ενημερώνεται με βάση το μοντέλο μετάβασης κατάστασης (το οποίο περιλαμβάνει και θόρυβο).

Ενημέρωση (Βάρη): Για κάθε σωματίδιο, υπολογίζεται η πιθανοφάνεια των παρατηρούμενων δεδομένων δεδομένης της κατάστασης του σωματιδίου χρησιμοποιώντας το μοντέλο παρατήρησης. Αυτές οι πιθανοφάνειες χρησιμοποιούνται για την ενημέρωση των βαρών των σωματιδίων.

Επανδειγματοληψία: Αν ο Αποτελεσματικός Αριθμός Δειγμάτων (ESS) των σωματιδίων πέσει κάτω από ένα όριο, πραγματοποιείται επανδειγματοληψία. Αυτή η διαδικασία περιλαμβάνει τη δημιουργία ενός νέου συνόλου σωματιδίων με δειγματοληψία από το τρέχον σύνολο με βάση τα βάρη τους.

Εκτίμηση: Η νέα θέση του αντικειμένου εκτιμάται ως ο σταθμισμένος μέσος όρος των καταστάσεων των σωματιδίων.

Βρόχος: Η διαδικασία επαναλαμβάνεται για κάθε frame στο βίντεο.

Λεπτομέρειες Υλοποίησης

Η υλοποίηση πραγματοποιήθηκε σε MATLAB. Η διαδικασία υλοποίησης περιλαμβάνει τα ακόλουθα κύρια βήματα:

Αρχικοποίηση Βίντεο

- **Ανάγνωση Βίντεο:** Το βίντεο αναγνωρίζεται χρησιμοποιώντας την εντολή VideoReader, η οποία επιτρέπει την ανάγνωση των frame από το αρχείο βίντεο sevenup.m4v.

- **Αποθήκευση Βίντεο:** Το εξαγόμενο βίντεο αποθηκεύεται χρησιμοποιώντας το αντικείμενο VideoWriter, το οποίο αρχικοποιείται στην αρχή του προγράμματος και κλείνει στο τέλος, μετά την επεξεργασία όλων των frame.

Αρχικοποίηση Σωματιδίων

- **Αρχική Θέση:** Η αρχική θέση του ορθογωνίου ορίζεται στο (i0, j0) και τα σωματίδια παράγονται γύρω από αυτή τη θέση χρησιμοποιώντας την κανονική κατανομή mvnrnd.
- **Αριθμός Σωματιδίων:** Χρησιμοποιούνται **N = 100 σωματίδια**, και οι αρχικές τους θέσεις προστίθενται γύρω από την αρχική θέση με τυχαία κατανομή σύμφωνα με τη συνδιακύμανση V.

Εντοπισμός Ακμών και Υπολογισμός Πιθανοφάνειας

- **Μάσκες:** Δημιουργείται μια μάσκα που περιορίζει την περιοχή ενδιαφέροντος σε ένα παράθυρο γύρω από την τρέχουσα θέση του ορθογωνίου, ώστε να επικεντρωθεί η διαδικασία εντοπισμού ακμών σε αυτή την περιοχή.
- **Εντοπισμός Ακμών:** Εφαρμόζεται ο ανιχνευτής ακμών Canny στον πίνακα της μάσκας και εντοπίζονται οι ακμές στην περιοχή αυτή. Επιπλέον, εφαρμόζεται φίλτρο Sobel για τον εντοπισμό κάθετων ακμών, και τα αποτελέσματα συνδυάζονται.
- **Μετασχηματισμός Απόστασης:** Χρησιμοποιείται η εντολή bwdist() για να υπολογιστεί η απόσταση κάθε pixel από την πλησιέστερη ακμή στην εικόνα των εντοπισμένων ακμών.
- **Πιθανοφάνεια:** Για κάθε σωματίδιο, η πιθανοφάνεια υπολογίζεται χρησιμοποιώντας τη μέση απόσταση των ακμών του ορθογωνίου από τις εντοπισμένες ακμές, σύμφωνα με το μοντέλο παρατήρησης $p(x|z)=e^{-a.d}$.

Επανδειγματοληψία

Τα σωματίδια επανδειγματοληπτούνται αν ο Αποτελεσματικός Αριθμός Δειγμάτων (ESS) πέσει κάτω από το μισό του συνολικού αριθμού σωματιδίων. Αυτή η διαδικασία διασφαλίζει ότι το σύνολο των σωματιδίων παραμένει διαφοροποιημένο και αποφεύγεται η εκφυλιστική διαδικασία.

Εκτίμηση Θέσης

Η τελική θέση του ορθογωνίου στο τρέχον frame υπολογίζεται ως ο σταθμισμένος μέσος όρος των θέσεων όλων των σωματιδίων. Αυτή η εκτίμηση χρησιμοποιείται για να ενημερωθεί η θέση του αντικειμένου στο επόμενο frame.

Σχεδίαση και Αποθήκευση

Η τρέχουσα θέση του ορθογωνίου σχεδιάζεται πάνω στο frame και το frame αποθηκεύεται στο εξαγόμενο βίντεο. Η διαδικασία επαναλαμβάνεται για όλα τα frame του βίντεο.

Αποτελέσματα και Παρατηρήσεις

Το σύστημα παρακολούθησης αποδείχθηκε ιδιαίτερα αποτελεσματικό στη διαχείριση της αβεβαιότητας στη θέση του αντικειμένου. Το φίλτρο σωματιδίων διαχειρίζεται τη θορυβώδη φύση των παρατηρήσεων, ενώ η χρήση του μοντέλου παρατήρησης βασισμένου σε ακμές εξασφαλίζει ακριβή εντοπισμό του αντικειμένου, ακόμη και σε σκηνές με θόρυβο.

Ο συνδυασμός των φίλτρων Canny και Sobel για τον εντοπισμό ακμών παρέχει ένα αποτελεσματικό μέσο για την ανίχνευση των σχετικών ακμών του αντικειμένου.

Το βήμα επανδειγματοληψίας εξασφαλίζει την ποικιλομορφία του συνόλου των σωματιδίων, καθιστώντας το σύστημα ανθεκτικό σε μεγάλες αποκλίσεις.

Το σύστημα μπορεί να παρακολουθεί το αντικείμενο ακόμη και όταν η ακριβής θέση του είναι δύσκολο να εντοπιστεί λόγω θορύβου ή κίνησης της κάμερας.

Ανάλυση των Μεταβλητών C, V, a και N στο Σύστημα Παρακολούθησης με Φίλτρο Σωματιδίων

Στο πλαίσιο του φίλτρου σωματιδίων που υλοποιείται για την παρακολούθηση ενός αντικειμένου σε βίντεο, οι μεταβλητές C, V, a και N παίζουν κρίσιμο ρόλο στον καθορισμό της συμπεριφοράς του φίλτρου και της απόδοσής του. Ας εξετάσουμε λεπτομερώς αυτές τις μεταβλητές:

Μεταβλητή C - Συντελεστής Θορύβου στην Κατάσταση (Noise Covariance Coefficient)

Η μεταβλητή C είναι ένας συντελεστής που επηρεάζει τη διακύμανση του θορύβου στη δυναμική του συστήματος, δηλαδή στον τρόπο που εξελίσσεται η θέση των σωματιδίων από frame σε frame. Η C καθορίζει την "ένταση" του θορύβου που προστίθεται στις θέσεις των σωματιδίων σε κάθε βήμα.

Ρόλος της C:

- **Ρύθμιση της Ελευθερίας Κίνησης των Σωματιδίων:** Όσο μεγαλύτερη είναι η τιμή της C, τόσο μεγαλύτερη είναι η διασπορά (dispersion) των σωματιδίων σε κάθε βήμα, επιτρέποντάς τους να εξερευνούν μεγαλύτερη περιοχή γύρω από την εκτιμώμενη θέση του αντικειμένου.
- **Αντιμετώπιση Αβεβαιότητας:** Αν το αντικείμενο κινείται απρόβλεπτα, μια μεγαλύτερη τιμή C επιτρέπει στα σωματίδια να προσαρμόζονται καλύτερα στις απότομες αλλαγές της θέσης του αντικειμένου.

Επιλογή της Τιμής της C:

- Αν η C είναι πολύ μικρή, τα σωματίδια δεν θα κινούνται αρκετά, με αποτέλεσμα το φίλτρο να "χάνει" το αντικείμενο αν αυτό μετακινείται γρήγορα.
- Αν η C είναι πολύ μεγάλη, τα σωματίδια μπορεί να διασκορπιστούν υπερβολικά, προκαλώντας απώλεια ακρίβειας στην εκτίμηση της θέσης του αντικειμένου.
- Στην υλοποίηση που παρουσιάζεται, επιλέχθηκε μια τιμή **C = 6**, η οποία επιτρέπει ένα ισορροπημένο επίπεδο διασποράς, ικανό να παρακολουθεί με ακρίβεια το αντικείμενο υπό φυσιολογικές συνθήκες κίνησης.

Μεταβλητή V - Συνδιακύμανση Θορύβου (Noise Covariance Matrix)

Η μεταβλητή V είναι ένας πίνακας συνδιακύμανσης (covariance matrix) που χρησιμοποιείται για να καθορίσει τη διασπορά και την κατεύθυνση του θορύβου που προστίθεται στην κίνηση των σωματιδίων σε κάθε βήμα.

Ρόλος της V:

- **Εξέλιξη Κατάστασης:** Ο πίνακας V διαμορφώνει τον τρόπο με τον οποίο τα σωματίδια διασπείρονται από το ένα frame στο άλλο. Διαφορετικές τιμές στον πίνακα αυτόν μπορούν να επηρεάσουν την απόσταση και την κατεύθυνση της κίνησης των σωματιδίων.

- **Αντιμετώπιση Ετερογένειας:** Αν το αντικείμενο κινείται περισσότερο σε μία κατεύθυνση (π.χ., οριζόντια), ο πίνακας V μπορεί να διαμορφωθεί ώστε να δίνει μεγαλύτερη διασπορά σε αυτή την κατεύθυνση.

Δομή της V: Ο πίνακας V είναι συμμετρικός, με τις διαγώνιες τιμές να αντιπροσωπεύουν τις διασπορές (variances) και τις εκτός διαγώνιες να αντιπροσωπεύουν τις συνδιακυμάνσεις (covariances) μεταξύ των διαστάσεων:

Επιλογή της V:

- **Διαγώνιες Τιμές (Variance):** Καθορίζουν το πόσο τα σωματίδια μπορούν να μετακινηθούν ανεξάρτητα σε κάθε κατεύθυνση.
- **Εκτός Διαγωνίων Τιμές (Covariance):** Καθορίζουν τη συσχέτιση μεταξύ των κινήσεων σε x και y.

Στην υλοποίηση, χρησιμοποιείται η εξής μορφή:

$$V = C \begin{bmatrix} 14 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Αυτό σημαίνει ότι η διασπορά είναι σχεδόν μηδενική στον κατακόρυφο άξονα και δεν υπάρχει συνδιακύμανση, που υποδηλώνει ότι οι κινήσεις σε x και y είναι τελείως ανεξάρτητες.

Μεταβλητή a - Συντελεστής Εξασθένησης Πιθανοφάνειας (Likelihood Decay Factor)

Η μεταβλητή a είναι ένας συντελεστής που χρησιμοποιείται στο μοντέλο παρατήρησης για να καθορίσει την ταχύτητα εξασθένησης της πιθανοφάνειας σε σχέση με την απόσταση d της εντοπισμένης ακμής από τις πλευρές του ορθογωνίου.

Ρόλος της a:

- **Πιθανοφάνεια Παρατήρησης**: Ο συντελεστής a ελέγχει το πόσο γρήγορα μειώνεται η πιθανοφάνεια p(x|z) όταν αυξάνεται η απόσταση d. Με μεγαλύτερη τιμή του a, η συνάρτηση πιθανοφάνειας γίνεται πιο απότομη, πράγμα που σημαίνει ότι οι θέσεις μακριά από τις ακμές έχουν πολύ μικρή πιθανοφάνεια.
- **Ευαισθησία στο Λάθος:** Μια μεγαλύτερη τιμή του a καθιστά το φίλτρο πιο ευαίσθητο στα σφάλματα στην εκτίμηση της θέσης, καθώς ακόμη και μικρές αποκλίσεις θα μειώσουν δραστικά την πιθανοφάνεια.

Συνάρτηση Πιθανοφάνειας:

Η συνάρτηση πιθανοφάνειας που χρησιμοποιείται είναι:

$$p(x|z)=e^{-a \cdot d}$$

όπου:

- **d** είναι η μέση απόσταση των ακμών του ορθογωνίου από τις εντοπισμένες ακμές,
- **a** είναι ο συντελεστής εξασθένησης.

Επιλογή της Τιμής της a:

- **Χαμηλή Τιμή a:** Το φίλτρο θα είναι πιο ανεκτικό σε μικρές αποκλίσεις, αλλά μπορεί να χάσει την ακρίβεια στην παρακολούθηση.
- **Υψηλή Τιμή a**: Το φίλτρο θα είναι πιο ακριβές, αλλά μπορεί να είναι πιο επιρρεπές σε θόρυβο ή μικρές ανακρίβειες.

Στην υλοποίηση, επιλέγεται μια τιμή **a = 2**, που παρέχει μια καλή ισορροπία μεταξύ ακρίβειας και αντοχής στον θόρυβο.

Μεταβλητή N – Πλήθος σωματιδίων

Η N αντιπροσωπεύει τον αριθμό των σωματιδίων που χρησιμοποιούνται στο φίλτρο σωματιδίων. Κάθε σωματίδιο είναι μια πιθανή εκτίμηση της θέσης του παρακολουθούμενου αντικειμένου, και η συνολική εκτίμηση της θέσης προκύπτει από τον σταθμισμένο μέσο όρο των θέσεων όλων των σωματιδίων.

Ρόλος της N:

- **Ακρίβεια Εκτίμησης:** Όσο περισσότερα σωματίδια χρησιμοποιούνται (δηλαδή, όσο μεγαλύτερη είναι η N), τόσο καλύτερη είναι η ικανότητα του φίλτρου να καλύπτει την περιοχή αναζήτησης και να παρέχει μια ακριβέστερη εκτίμηση της θέσης του αντικειμένου.
- **Ανθεκτικότητα στον Θόρυβο:** Με περισσότερα σωματίδια, το φίλτρο γίνεται πιο ανθεκτικό στον θόρυβο, καθώς είναι πιο πιθανό κάποια σωματίδια να βρίσκονται κοντά στη σωστή θέση του αντικειμένου ακόμα και όταν υπάρχει θόρυβος στα δεδομένα.
- **Αντιμετώπιση Αβεβαιότητας:** Σε καταστάσεις όπου το αντικείμενο μετακινείται απρόβλεπτα ή όταν υπάρχουν απότομες αλλαγές στην κατεύθυνση ή την ταχύτητα, ένα μεγαλύτερο N βοηθά το φίλτρο να προσαρμόζεται καλύτερα σε αυτές τις αλλαγές.

Επιλογή της Τιμής της N:

Η επιλογή της τιμής της N αποτελεί κρίσιμο βήμα στη ρύθμιση του φίλτρου σωματιδίων, καθώς επηρεάζει άμεσα την απόδοση και την ταχύτητα της διαδικασίας παρακολούθησης.

Υψηλή Τιμή N:

- Πλεονεκτήματα:
    - Καλύτερη κάλυψη του χώρου αναζήτησης.
    - Μεγαλύτερη ακρίβεια στην εκτίμηση της θέσης του αντικειμένου.
    - Περισσότερα σωματίδια σημαίνουν περισσότερες πιθανότητες να εντοπιστεί η σωστή θέση, ακόμα και σε περιπτώσεις με πολύ θόρυβο ή γρήγορες κινήσεις του αντικειμένου.
- Μειονεκτήματα:
    - Αυξημένη υπολογιστική επιβάρυνση, καθώς απαιτείται περισσότερος χρόνος για τον υπολογισμό των πιθανοτήτων και την ανανέωση των θέσεων των σωματιδίων.
    - Αύξηση της πολυπλοκότητας του συστήματος.

Επιλογή για την Υλοποίηση:

Στην παρούσα υλοποίηση, επιλέγεται **N = 100** σωματίδια.

Αυτή η τιμή παρέχει έναν καλό συμβιβασμό μεταξύ της ακρίβειας και της υπολογιστικής επιβάρυνσης, επιτρέποντας στο σύστημα να λειτουργεί αποτελεσματικά και να παρακολουθεί το αντικείμενο με ικανοποιητική ακρίβεια χωρίς να απαιτείται υπερβολικός υπολογιστικός χρόνος.

Σχέση της N με το Αποτελεσματικό Μέγεθος Δείγματος (Effective Sample Size - ESS)

Η μεταβλητή N συνδέεται άμεσα με το αποτελεσματικό μέγεθος δείγματος (ESS), το οποίο μετρά την ποικιλία των σωματιδίων μετά τη στάθμιση. Αν το ESS είναι χαμηλό, σημαίνει ότι μόνο λίγα σωματίδια έχουν σημαντικό βάρος, και συνεπώς το φίλτρο δεν λειτουργεί αποτελεσματικά.

Συμπεράσματα

Οι μεταβλητές C, V, N και a είναι ζωτικής σημασίας για την επιτυχημένη υλοποίηση και απόδοση του φίλτρου σωματιδίων. Η επιλογή τους εξαρτάται από τη φύση του αντικειμένου που παρακολουθείται, τον τύπο του θορύβου στα δεδομένα, καθώς και τις απαιτήσεις σε ακρίβεια και ανθεκτικότητα του συστήματος. Η σωστή ρύθμιση αυτών των παραμέτρων μπορεί να οδηγήσει σε ένα φίλτρο σωματιδίων που είναι ακριβές και αποτελεσματικό στην παρακολούθηση αντικειμένων σε δυναμικά περιβάλλοντα.

2 Προσπαθήστε να επιλύσετε το πρόβλημα επεκτείνοντας το μοντέλο σας ώστε να συμπεριλάβετε τον προσανατολισμό, που θα αναπαριστά με τη περιστροφή του κουτιού και χρησιμοποιήστε το στην ακολουθία coke.

Στο πλαίσιο της παρακολούθησης αντικειμένων μέσω φίλτρου σωματιδίων, έγιναν σημαντικές τροποποιήσεις στον αρχικό κώδικα που εφαρμόζει την τεχνική αυτή. Οι κύριες αλλαγές αφορούν την προσθήκη παρακολούθησης του προσανατολισμού και την αντικατάσταση του βίντεο με μια ακολουθία εικόνων. Παρακάτω αναλύονται οι κύριες διαφορές:

1. **Επεξεργασία Εικόνας**
   Στον αρχικό κώδικα, χρησιμοποιείται βίντεο ως είσοδος, ενώ στον νέο κώδικα χρησιμοποιείται μια ακολουθία εικόνων. Αυτό επέβαλε τη χρήση της εντολής imread για την ανάγνωση εικόνων αντί του readFrame για την ανάγνωση frame από το βίντεο. Επίσης, οι εικόνες φορτώνονται από έναν συγκεκριμένο φάκελο και η ανάλυση παραμένει σταθερή σε 480x640.
2. **Παρακολούθηση Προσανατολισμού**
   Μία από τις πιο σημαντικές προσθήκες είναι η εισαγωγή του προσανατολισμού (γωνία περιστροφής) ως παράμετρο παρακολούθησης. Πιο συγκεκριμένα:
   - Ο αρχικός κώδικας χρησιμοποιούσε μόνο τις συντεταγμένες i,j για την παρακολούθηση της θέσης του αντικειμένου. Στον νέο κώδικα, έχει προστεθεί και η παράμετρος θ για τον προσανατολισμό.

- Η συνάρτηση plotRotatedRectangle χρησιμοποιείται για την απεικόνιση του αντικειμένου με περιστροφή κατά την γωνία θ.

3. **Στατιστικό Μοντέλο και Διασπορά**

Το μοντέλο θορύβου στο νέο κώδικα επεκτάθηκε ώστε να περιλαμβάνει την παράμετρο του προσανατολισμού. Ο πίνακας συνδιακύμανσης V είναι τώρα 3x3 και περιλαμβάνει διασπορές και συνδιακυμάνσεις μεταξύ των συντεταγμένων i, j, και της γωνίας θ. Η μεταβολή αυτή επιτρέπει μεγαλύτερη ευελιξία και ακρίβεια στην παρακολούθηση, αφού λαμβάνεται υπόψη και η γωνιακή απόκλιση.

$$V = C \begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix} = C \begin{matrix} 45 & 15 & -0.25 \\ 15 & 50 & 1 \\ -0.25 & 1 & 0.7 \end{matrix}$$

Διαγώνια Στοιχεία (a, e, i):

- a: Διασπορά στη θέση x.
- e: Διασπορά στη θέση y.
- i: Διασπορά στον προσανατολισμό θ.
-

Αυτές οι τιμές καθορίζουν πόσο μπορούν να κινηθούν ή να περιστραφούν τα σωματίδια λόγω θορύβου. Μεγαλύτερες τιμές σημαίνουν μεγαλύτερη διασπορά επιτρέποντας στα σωματίδια να κινηθούν ή να περιστραφούν πιο ελεύθερα.

Εκτός Διαγωνίου Στοιχεία (b, c, d, f, g, h):

- b και d: Συνδιακύμανση μεταξύ x και y.
- c και g: Συνδιακύμανση μεταξύ x και θ.
- f και h: Συνδιακύμανση μεταξύ y και θ.

Αυτές οι τιμές αντιπροσωπεύουν τη σχέση μεταξύ ζευγών μεταβλητών. Αν είναι θετικές, οι μεταβλητές συσχετίζονται θετικά, αν είναι αρνητικές, συσχετίζονται αρνητικά

4. **Υπολογισμός Πιθανοτήτων και Αναδειγματοληψία**

Οι πιθανότητες παρατήρησης για κάθε σωματίδιο στον νέο κώδικα υπολογίζονται λαμβάνοντας υπόψη την απόσταση από το αντικείμενο και τον προσανατολισμό του. Η διαδικασία αναδειγματοληψίας προσαρμόστηκε επίσης για να συμπεριλάβει την νέα παράμετρο θ.

5. **Οπτικοποίηση**

Ο νέος κώδικας περιλαμβάνει την οπτικοποίηση του περιγράμματος του αντικειμένου με περιστροφή, το οποίο είναι κρίσιμο για την παρακολούθηση αντικειμένων που μπορεί να αλλάζουν γωνία κατά την κίνηση. Αυτή η τροποποίηση βελτιώνει σημαντικά την ακρίβεια της απεικόνισης.

Αυτές οι αλλαγές βελτιώνουν την ικανότητα του αλγορίθμου να παρακολουθεί αντικείμενα σε περιβάλλοντα όπου το αντικείμενο μπορεί να αλλάξει γωνία ή κατεύθυνση κατά την κίνηση, καθιστώντας το σύστημα πιο αξιόπιστο και ακριβές σε πιο απαιτητικές συνθήκες παρακολούθησης.

3 (Προαιρετικό) Υλοποιήστε τα ερωτήματα 1,2 με χρήση φίλτρου Kalman.

Το φίλτρο Kalman αποτελεί ένα από τα πιο ισχυρά και δημοφιλή αλγοριθμικά εργαλεία στην επιστήμη της μηχανικής μάθησης και της επεξεργασίας σήματος. Αναπτύχθηκε από τον Rudolf Kalman το 1960 και χρησιμοποιείται κυρίως για την εκτίμηση της κατάστασης ενός συστήματος δυναμικού που εξελίσσεται με την πάροδο του χρόνου, ακόμα και όταν οι παρατηρήσεις του συστήματος είναι θορυβώδεις ή ατελείς. Το φίλτρο Kalman λειτουργεί σε πραγματικό χρόνο, επιτρέποντας τη συνεχή εκτίμηση της κατάστασης ενός συστήματος καθώς λαμβάνονται νέα δεδομένα.

3.1 Θεωρία του Φίλτρου Kalman

Το φίλτρο Kalman βασίζεται σε ένα μαθηματικό μοντέλο του συστήματος που περιλαμβάνει δύο βασικές εξισώσεις:

1. **Εξίσωση Κατάστασης (State Equation):** Περιγράφει πώς εξελίσσεται η κατάσταση του συστήματος με την πάροδο του χρόνου.
2. **Εξίσωση Μέτρησης (Measurement Equation):** Περιγράφει πώς σχετίζεται η παρατηρούμενη μέτρηση με την κατάσταση του συστήματος.

Εξίσωση Κατάστασης

Η εξίσωση κατάστασης έχει τη μορφή:

$$X_{k+1} = Ax_k + Bu_k + w_k$$

όπου:

$\mathbf{x_k}$ είναι το διάνυσμα της κατάστασης στο χρόνο k,

A είναι το μητρώο μετάβασης της κατάστασης που περιγράφει τη δυναμική του συστήματος,

B είναι το μητρώο ελέγχου που πολλαπλασιάζει το διάνυσμα ελέγχου u_k,

$\mathbf{w_k}$ είναι ο θόρυβος της διαδικασίας, ο οποίος θεωρείται ότι έχει κανονική κατανομή με μέση τιμή μηδέν και συνδιασπορά Q.

Εξίσωση Μέτρησης

Η εξίσωση μέτρησης έχει τη μορφή:

$$z_k = Hx_k + u_k$$

όπου:

$\mathbf{z_k}$ είναι το διάνυσμα των μετρήσεων στο χρόνο k,

H είναι το μητρώο μέτρησης που συσχετίζει την κατάσταση του συστήματος με τις μετρήσεις,

$\mathbf{u_k}$ είναι ο θόρυβος της μέτρησης, ο οποίος θεωρείται ότι έχει κανονική κατανομή με μέση τιμή μηδέν και συνδιασπορά R.

Βήματα του Φίλτρου Kalman

Το φίλτρο Kalman λειτουργεί μέσω δύο φάσεων: της πρόβλεψης και της διόρθωσης.

1. **Φάση Πρόβλεψης (Prediction Step):**
    - Προβλέπεται η νέα κατάσταση του συστήματος χρησιμοποιώντας την εξίσωση κατάστασης.
    - Υπολογίζεται η προβλεπόμενη συνδιασπορά του σφάλματος.
2. **Φάση Διόρθωσης (Correction Step):**
    - Λαμβάνονται υπόψη οι νέες μετρήσεις και διορθώνεται η πρόβλεψη με βάση τη διαφορά μεταξύ της πραγματικής μέτρησης και της προβλεπόμενης τιμής.
    - Ενημερώνεται η συνδιασπορά του σφάλματος.

3.2 Φίλτρο Kalman σε μια διάσταση

Ο MATLAB κώδικας χρησιμοποιεί το φίλτρο Kalman για την παρακολούθηση ενός αντικειμένου σε ένα βίντεο.

- **Ανάγνωση Βίντεο και Υπολογισμός Background:** Το βίντεο αναλύεται και υπολογίζεται η εικόνα του υποβάθρου, η οποία είναι ο μέσος όρος των πρώτων 90 frame. Το υπόβαθρο είναι σημαντικό για να προσδιοριστεί η αλλαγή των εικονοστοιχείων στην επόμενη φάση.

- **Αρχικοποίηση Παραμέτρων και Μεταβλητών:** Εδώ, αρχικοποιούνται οι παράμετροι για το φίλτρο Kalman
    - **dt (Χρονικό Βήμα):** Ορίζεται ως **1**, που αντιστοιχεί σε ένα frame ανά δευτερόλεπτο.
    - **R (Συνδιασπορά Θορύβου Μέτρησης):** Ορίζεται σε **0.01**, που σημαίνει ότι οι μετρήσεις είναι σχετικά ακριβείς με λίγο θόρυβο.
    - **Q (Συνδιασπορά Θορύβου Διαδικασίας):** Το μητρώο Q καθορίζει την αβεβαιότητα στη μοντελοποίηση του συστήματος, με μεγαλύτερη αβεβαιότητα στην ταχύτητα παρά στη θέση. $Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix}$
    - **P (Αρχική Συνδιασπορά Σφάλματος):** Αρχικά, θεωρείται ίση για τη θέση και την ταχύτητα. $P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
    - **A (Μητρώο Μετάβασης Κατάστασης):** Περιγράφει πώς η θέση και η ταχύτητα εξελίσσονται με την πάροδο του χρόνου. $A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$
    - **H (Μητρώο Μέτρησης):** Επειδή μετράμε μόνο τη θέση στον άξονα x, ορίζεται ως $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$.
- **Βρόχος tracking:** Υλοποιεί το φίλτρο Kalman, προβλέποντας τη νέα θέση και στη συνέχεια διορθώνοντας την πρόβλεψη με βάση τις μετρήσεις από το βίντεο.
    - **Ανάλυση Διαφοράς:** Υπολογίζεται η διαφορά μεταξύ της τρέχουσας εικόνας και του υποβάθρου για να εντοπιστούν οι αλλαγές (αντικείμενο).
    - **Εκτίμηση της Θέσης του Αντικειμένου:** Χρησιμοποιείται το φίλτρο Kalman για να εκτιμηθεί η νέα θέση του αντικειμένου, ακόμα και αν το αντικείμενο δεν εντοπιστεί.
    - **Διόρθωση της Πρόβλεψης:** Εάν εντοπιστεί το αντικείμενο, η πρόβλεψη διορθώνεται με βάση τη νέα μέτρηση.
    - **Οπτικοποίηση:** Η θέση του αντικειμένου απεικονίζεται σε κάθε frame με ένα κόκκινο πλαίσιο.

Το φίλτρο Kalman χρησιμοποιείται ευρέως για την παρακολούθηση αντικειμένων σε θορυβώδη περιβάλλοντα. Ο κώδικας που δημιουργήθηκε εφαρμόζει αυτήν την τεχνική στην ανάλυση ενός βίντεο, με ιδιαίτερη έμφαση στην παρακολούθηση της θέσης ενός αντικειμένου στον άξονα x. Η επιλογή των παραμέτρων, όπως το χρονικός βήμα, η συνδιασπορά του θορύβου μέτρησης και της διαδικασίας, επηρεάζει σημαντικά την απόδοση του φίλτρου και απαιτεί προσεκτική ρύθμιση για τη βελτιστοποίηση των αποτελεσμάτων.

Σύγκριση Φίλτρου Σωματιδίων και Φίλτρου Kalman

1. Αρχή Λειτουργίας

- **Φίλτρο Σωματιδίων:** Βασίζεται σε ένα σύνολο από σωματίδια που αναπαριστούν πιθανές καταστάσεις του συστήματος. Κάθε σωματίδιο έχει μια πιθανότητα που αντιστοιχεί στο πόσο πιθανό είναι να βρίσκεται το σύστημα σε αυτήν την κατάσταση. Το φίλτρο σωματιδίων χρησιμοποιεί τεχνικές Monte Carlo για την εκτίμηση της κατανομής πιθανότητας του συστήματος.
- **Φίλτρο Kalman:** Βασίζεται σε μια γραμμική αναδρομική μέθοδο που υποθέτει ότι το σύστημα ακολουθεί μια κανονική κατανομή (Gaussian). Το φίλτρο Kalman χρησιμοποιεί τις εξισώσεις πρόβλεψης και ενημέρωσης για να παρέχει μια εκτίμηση της κατάστασης του συστήματος και της αβεβαιότητάς της.

2. Γραμμικότητα και Μη Γραμμικότητα

- **Φίλτρο Σωματιδίων:** Είναι κατάλληλο για μη γραμμικά και μη Γκαουσσιανά συστήματα. Δεν υπάρχει ανάγκη για γραμμικότητα, και μπορεί να αντιμετωπίσει πολύπλοκες κατανομές πιθανότητας.
- **Φίλτρο Kalman:** Είναι βέλτιστο για γραμμικά και Γκαουσσιανά συστήματα. Για μη γραμμικά συστήματα, χρησιμοποιείται μια τροποποιημένη έκδοση, όπως το Extended Kalman Filter (EKF) ή το Unscented Kalman Filter (UKF), αλλά αυτές οι εκδοχές μπορούν να είναι λιγότερο ακριβείς σε περιπτώσεις πολύπλοκων μη γραμμικών συστημάτων.

3. Υπολογιστική Πολυπλοκότητα

- **Φίλτρο Σωματιδίων:** Έχει υψηλότερη υπολογιστική πολυπλοκότητα, ειδικά καθώς αυξάνεται ο αριθμός των σωματιδίων. Κάθε σωματίδιο απαιτεί ξεχωριστό υπολογισμό για την ενημέρωση της κατάστασής του, κάτι που μπορεί να είναι χρονοβόρο, ειδικά για μεγάλα συστήματα.
- **Φίλτρο Kalman:** Είναι υπολογιστικά πιο αποδοτικό. Οι υπολογισμοί είναι αναδρομικοί και μπορούν να γίνουν γρήγορα, ακόμα και για μεγάλα συστήματα. Γι' αυτό το φίλτρο Kalman χρησιμοποιείται ευρέως σε εφαρμογές πραγματικού χρόνου.

4. Προσαρμοστικότητα και Ευελιξία

- **Φίλτρο Σωματιδίων:** Προσφέρει μεγαλύτερη ευελιξία και μπορεί να προσαρμοστεί σε ένα ευρύ φάσμα καταστάσεων και μοντέλων, ανεξάρτητα από τη μορφή της κατανομής πιθανότητας. Μπορεί να διαχειριστεί πολυτροπικές κατανομές και να παρακολουθήσει περίπλοκες κινήσεις.
- **Φίλτρο Kalman:** Είναι πιο περιορισμένο στη χρήση του, καθώς βασίζεται σε γραμμικά μοντέλα και Γκαουσσιανές κατανομές. Αν και οι επεκτάσεις του φίλτρου Kalman μπορούν να αντιμετωπίσουν μη γραμμικά συστήματα, η απόδοσή τους μειώνεται σημαντικά σε περίπτωση σοβαρών μη γραμμικοτήτων.

5. Ακρίβεια και Αποτελεσματικότητα

- **Φίλτρο Σωματιδίων:** Μπορεί να προσφέρει υψηλότερη ακρίβεια σε μη γραμμικά και περίπλοκα συστήματα, ιδιαίτερα όταν η κατανομή πιθανότητας δεν είναι Γκαουσσιανή. Η ακρίβεια του φίλτρου εξαρτάται από τον αριθμό των σωματιδίων.
- **Φίλτρο Kalman:** Είναι εξαιρετικά αποτελεσματικό και ακριβές για γραμμικά συστήματα με Γκαουσσιανές αβεβαιότητες. Ωστόσο, η ακρίβειά του μειώνεται σε μη γραμμικά συστήματα, ιδιαίτερα εάν οι τροποποιήσεις όπως το EKF δεν είναι επαρκείς.

Συμπεράσματα

Το **Φίλτρο Σωματιδίων** είναι ιδανικό για πολύπλοκα, μη γραμμικά συστήματα όπου η κατανομή της πιθανότητας μπορεί να είναι αυθαίρετη και μη Γκαουσσιανή. Είναι όμως πιο απαιτητικό υπολογιστικά.

Το **Φίλτρο Kalman** προσφέρει εξαιρετική απόδοση σε γραμμικά συστήματα και χρησιμοποιείται ευρέως λόγω της υπολογιστικής του αποδοτικότητας, αλλά μπορεί να μην είναι κατάλληλο για περίπλοκα, μη γραμμικά συστήματα.

4 Κώδικας

4.1 Αλγόριθμος tracking με φίλτρο σωματιδίων

```
function tracking_with_particle_filter
clear;clc;close all;
%% Video Reader and Writer Initialization
videoFile = 'sevenup.m4v';
videoReader = VideoReader(videoFile);
videoWriter = VideoWriter('outputvideo1.avi');
open(videoWriter);
%% Parameters Setup
% Size of the video frames
DIM = 600;
% Size of the rectangle to track [width, height]
rectSize = [185, 455];
% Coefficient for the noise covariance(bigger C => higher dispersion of particles)
C = 6;
% Initial position of the rectangle
i0 = 455; j0 = 320;
% Mean and variance for the noise model
M = [0 0]';
% Diagonal elements represent the variances (spread) in x and y,
% Off-diagonal elements represent the covariance
V = C * [14 0; 0 0.01];
% Number of particles
N = 100;
%% Particle Initialization
% Initialize particles (add noise)
particles = mvnrnd(M, V, N) + repmat([i0 j0], N, 1);
% Initialize particle weights to the same value
w = ones(1, N) / N;
```

```
frameCount = 0;
while hasFrame(videoReader)
    %% Read and Process Frame
    frame = readFrame(videoReader);
    % Resize the frame
    resizedFrame = imresize(frame, [DIM, DIM]);
    % Convert frame to grayscale
    grayFrame = rgb2gray(resizedFrame);

    % Create a mask for the original black rectangle region
    mask = false(DIM, DIM);
    i1 = max(round(i0 - rectSize(1)/2), 1);
    i2 = min(round(i0 + rectSize(1)/2), DIM);
    j1 = max(round(j0 - rectSize(2)/2), 1);
    j2 = min(round(j0 + rectSize(2)/2), DIM);
    mask(j1:j2, i1:i2) = true;

    % Apply edge detection only within the masked region
    grayFrameMasked = grayFrame;
    grayFrameMasked(~mask) = 0;

    % Apply Canny edge detector
    cannyEdges = edge(grayFrameMasked, 'Canny');
    % Apply Sobel filter to detect vertical edges
    sobelVerticalEdges = edge(grayFrameMasked, 'Sobel', 'vertical');
    % Combine Canny edges with vertical edges
    combinedEdges = cannyEdges & sobelVerticalEdges;
    % Distance transform of the inverse combined edge image
    D = bwdist(~combinedEdges);

    %% Likelihood Calculation
    threshold = 1e-20;
    % Get the likelihood for each particle
    for c = 1:N
        w(c) = calculateObservationLikelihood(rectSize, D, particles(c, 1),
particles(c, 2));
        if w(c) < threshold
            w(c) = 0;
        end
    end
    %% Weight normalization
    if sum(w) == 0
        break;
    end
    w = w / sum(w);

    %% Effective Sample Size (ESS)
    ess = 1 / sum(w.^2);
    if ess < N / 2
        % Resample particles based on their weights
        new_particles = resample_particles(particles, w, N, M, V);
        particles = new_particles;
        w = ones(1, N) / N;
    end
    %% Create Image for Plotting Particles
    % Draw all particles into an RGB image with red color
    particle_image = zeros(DIM, DIM);
    for c = 1:N
        particle_image = particle_image + plotRectangle(rectSize, particles(c, 1),
particles(c, 2), DIM);
    end
    mask = (particle_image > 1);
    particle_image(mask) = 1;
    % Create the final image by overlaying particles on the original frame
```

```
    total_image = im2double(resizedFrame);
    particle_image_rgb = zeros(DIM, DIM, 3);
    particle_image_rgb(:, :, 1) = particle_image; % Red channel
    mask_particles = (particle_image_rgb > 0);
    total_image(mask_particles) = particle_image_rgb(mask_particles);

    %% Resampling Particles
    % Do resampling based on the weights
    for c = 1:N
        s = performSampling(w);
        M1 = [particles(s, 1); particles(s, 2)] + M;
        y(c,:) = M1 + mvnrnd(M,V,1)';
    end
    particles = y;

    %% Draw the target into an RGB image with black color (and set background to
white)
    mask_t = (combinedEdges == 1);
    mask = (total_image(:, :, 1) == 0 & total_image(:, :, 2) == 0 & total_image(:, :,
3) == 0);
    for c = 1:3
        x = total_image(:, :, c);
        x(mask) = 1;
        x(mask_t) = 0;
        total_image(:, :, c) = x;
    end

    %% Calculate New Position and Draw Thick Line
    % Calculate the new position of the rectangle (target object)
    i0 = sum(particles(:, 1) .* w');
    j0 = sum(particles(:, 2) .* w');
    % Draw the target object (black rectangle) in the current frame
    total_image = plotRectangleOutline(total_image,rectSize,i0,j0,[0 0 0]);
    % Display the current frame with tracked particles
    imshow(total_image);
    drawnow;
    % Write the frame with tracked particles to the output video
    writeVideo(videoWriter, im2frame(im2uint8(total_image)));
    % Increment frame counter
    frameCount = frameCount + 1;
end
close(videoWriter);
end

%% Calculate likelihood of a particle based on the distance from the real center
function out = calculateObservationLikelihood(rectSize, D, i, j)
% Decay factor
a = 2;
% The particle rectangle image
particle_image = plotRectangle(rectSize, i, j, size(D, 1));
f = exp(-a * D);
out = sum(sum(particle_image .* f));
end

function new_particles = resample_particles(particles, w, N, M, V)
    new_particles = zeros(size(particles));
    cumulative_sum = cumsum(w);
    for i = 1:N
        random_number = rand;
        index = find(cumulative_sum >= random_number, 1);
        new_particles(i, :) = particles(index, :) + mvnrnd(M, V, 1);
    end
end
```

```
function outimage = plotRectangle(rectSize, i0, j0, DIM)
w = rectSize(1);
h = rectSize(2);

i1 = max(round(i0 - w/2), 1);
i2 = min(round(i0 + w/2), DIM);
j1 = max(round(j0 - h/2), 1);
j2 = min(round(j0 + h/2), DIM);

outimage = zeros(DIM, DIM);
outimage(j1:j2, i1:i2) = 1;
end

function outimage = plotRectangleOutline(image, rectSize, i0, j0, color)
w = rectSize(1);
h = rectSize(2);

i1 = max(round(i0 - w/2), 1);
i2 = min(round(i0 + w/2), size(image, 2));
j1 = max(round(j0 - h/2), 1);
j2 = min(round(j0 + h/2), size(image, 1));

% Draw rectangle outline using line functions
for c = 1:3
    channel = image(:,:,c);
    % Top edge
    channel(j1, i1:i2) = color(c);
    % Bottom edge
    channel(j2, i1:i2) = color(c);
    % Left edge
    channel(j1:j2, i1) = color(c);
    % Right edge
    channel(j1:j2, i2) = color(c);
    image(:,:,c) = channel;
end

outimage = image;
end
```

4.2 Αλγόριθμος tracking με φίλτρο σωματιδίων & προσανατολισμός

```
function tracking_with_particle_filter_orientations
clear;clc;close all;
%% File and Folder Setup
imageFolder = 'Coke/img';  % Folder containing the images
imageFiles = dir(fullfile(imageFolder, '*.jpg'));  % List all jpg files in the folder
numImages = length(imageFiles);  % Number of images in the folder
groundTruthFile = 'Coke/groundtruth_rect.txt';
% Read ground truth positions and sizes
groundTruth = readmatrix(groundTruthFile);
%% Video Writer Initialization
videoWriter = VideoWriter('outputvideo2.avi');
open(videoWriter);
%% Parameters Setup
DIM1 = 480;  % Height
DIM2 = 640;  % Width
% Size of the rectangle to track [width, height]
rectSize = groundTruth(1, 3:4);  % Initial width and height from the first ground
truth entry
% Coefficient for the noise covariance (bigger C => higher dispersion of particles)
C = 6;
%% final
```

```
% Initial position of the rectangle
i0 = groundTruth(1, 1) + rectSize(1)/2;  % x-coordinate of the center from ground
truth
j0 = groundTruth(1, 2) + rectSize(2)/2;  % y-coordinate of the center from ground
truth
theta0 = 0;  % Initial orientation angle in degrees

% Mean and variance for the noise model
M = [0 0 0]';
V = C * [45 15 -0.25;...
         15 50 1;...
        -0.25 1 0.7];
% Number of particles
N = 100;
%% Particle Initialization
% Initialize particles (position + orientation)
particles = mvnrnd(M, V, N) + repmat([i0 j0 theta0], N, 1);
% Initialize particle weights to the same value
w = ones(1, N) / N;
%% Process Each Image
for frameCount = 1:numImages
    %% Read and Process Frame
    imageFile = fullfile(imageFolder, imageFiles(frameCount).name);
    frame = imread(imageFile);
    % Convert frame to grayscale
    grayFrame = rgb2gray(frame);

    % Create a mask for the original black rectangle region
    mask = false(DIM1, DIM2);
    i1 = max(round(i0 - rectSize(1)/2), 1);
    i2 = min(round(i0 + rectSize(1)/2), DIM1);
    j1 = max(round(j0 - rectSize(2)/2), 1);
    j2 = min(round(j0 + rectSize(2)/2), DIM2);
    mask(j1:j2, i1:i2) = true;

    % Apply edge detection only within the masked region
    grayFrameMasked = grayFrame;
    grayFrameMasked(~mask) = 0;

    % Apply Canny edge detector
    cannyEdges = edge(grayFrameMasked, 'Canny');
    combinedEdges = cannyEdges;
    % Distance transform of the inverse combined edge image
    D = bwdist(~combinedEdges);

    %% Likelihood Calculation
    threshold = 1e-20;
    % Get the likelihood for each particle
    for c = 1:N
        w(c) = calculateObservationLikelihood(rectSize, D, particles(c, 1),
particles(c, 2), particles(c, 3), DIM1,DIM2);
        if w(c) < threshold
            w(c) = 0;
        end
    end
    %% Weight normalization
    if sum(w) == 0
        break;
    end
    w = w / sum(w);

    %% Effective Sample Size (ESS)
    ess = 1 / sum(w.^2);
    if ess < N / 2
```

```
        % Resample particles based on their weights
        new_particles = resample_particles(particles, w, N, M, V);
        particles = new_particles;
        w = ones(1, N) / N;
    end

    %% Create Image for Plotting Particles
    % Draw all particles into an RGB image with red color
    particle_image = zeros(DIM1, DIM2);
    for c = 1:N
        particle_image = particle_image + plotRotatedRectangle(rectSize, particles(c,
1), particles(c, 2), particles(c, 3),DIM1,DIM2);
    end
    mask = (particle_image > 1);
    particle_image(mask) = 1;
    % Create the final image by overlaying particles on the original frame
    total_image = im2double(frame);
    particle_image_rgb = zeros(DIM1, DIM2, 3);
    particle_image_rgb(:, :, 1) = particle_image; % Red channel
    mask_particles = (particle_image_rgb > 0);
    total_image(mask_particles) = particle_image_rgb(mask_particles);

    %% Resampling Particles
    % Resample particles based on their weights and add noise
    for c = 1:N
        s = performSampling(w);
        M1 = [particles(s, 1); particles(s, 2); particles(s, 3)] + M;
        % Add noise to x and y only
        y(c,:) = M1 + mvnrnd(M,V,1)';
    end
    particles = y;

    %% Draw the target into an RGB image with black color (and set background to
white)
    mask_t = (combinedEdges == 1);
    mask = (total_image(:, :, 1) == 0 & total_image(:, :, 2) == 0 & total_image(:, :,
3) == 0);
    for c = 1:3
        x = total_image(:, :, c);
        x(mask) = 1;
        x(mask_t) = 0;
        total_image(:, :, c) = x;
    end

    %% Calculate New Position and Draw Thick Line
    % Calculate the new position and orientation of the rectangle (target object)
    i0 = sum(particles(:, 1) .* w');
    j0 = sum(particles(:, 2) .* w');
    theta0 = sum(particles(:, 3) .* w');

    % Draw the target object (rotated black rectangle) in the current frame
    total_image = plotRectangleOutline(total_image, rectSize, i0, j0,theta0, [0 0 0]);

    % Draw the ground truth rectangle in green
    gt_i0 = groundTruth(frameCount, 1) + rectSize(1) / 2;
    gt_j0 = groundTruth(frameCount, 2) + rectSize(2) / 2;
    total_image = plotRectangleOutline(total_image, rectSize, gt_i0, gt_j0,theta0, [0
1 0]);


    % Display the current frame with tracked particles
    imshow(total_image);
    drawnow;
    % Write the frame with tracked particles to the output video
```

```
    writeVideo(videoWriter, im2frame(im2uint8(total_image)));
end

close(videoWriter);
end

%% Calculate likelihood of a particle based on the distance from the real center
function out = calculateObservationLikelihood(rectSize, D, i, j,theta,DIM1,DIM2)
% Decay factor
a = 2;
% The particle rectangle image, rotated by theta
particle_image = plotRotatedRectangle(rectSize, i, j,theta, DIM1,DIM2);
f = exp(-a * D);
out = sum(sum(particle_image .* f));
end
%%
function new_particles = resample_particles(particles, w, N, M, V)
    new_particles = zeros(size(particles));
    cumulative_sum = cumsum(w);
    for i = 1:N
        random_number = rand;
        index = find(cumulative_sum >= random_number, 1);
        new_particles(i, 1:3) = particles(index, 1:3) + mvnrnd(M, V, 1);
    end
end
%%
function outimage = plotRotatedRectangle(rectSize, i0, j0, theta, DIM1, DIM2)
    w = rectSize(1);
    h = rectSize(2);

    % Calculate the top-left and bottom-right corners of the rectangle
    i1 = max(round(i0 - w/2), 1);
    i2 = min(round(i0 + w/2), DIM2);
    j1 = max(round(j0 - h/2), 1);
    j2 = min(round(j0 + h/2), DIM1);

    % Create a mask with a filled rectangle
    rect = zeros(DIM1, DIM2);
    rect(j1:j2, i1:i2) = 1;

    % Rotate the rectangle
    outimage = imrotate(rect, theta, 'crop');
end

%%
function outimage = plotRectangleOutline(image, rectSize, i0, j0, theta, color)
    w = rectSize(1);
    h = rectSize(2);

    % Calculate the four corners of the rectangle before rotation
    corners = [
        -w/2, -h/2;
         w/2, -h/2;
         w/2,  h/2;
        -w/2,  h/2
    ];

    % Rotation matrix
    R = [cosd(theta), -sind(theta); sind(theta), cosd(theta)];

    % Rotate and translate corners
    rotated_corners = (R * corners')' + [i0, j0];

    % Draw lines between the corners
```

```
    outimage = image;
    for k = 1:4
        x1 = rotated_corners(k, 1);
        y1 = rotated_corners(k, 2);
        x2 = rotated_corners(mod(k, 4) + 1, 1);
        y2 = rotated_corners(mod(k, 4) + 1, 2);
        outimage = insertShape(outimage, 'Line', [x1, y1, x2, y2], 'Color', color,
'LineWidth', 2);
    end
end
%%
function index = performSampling(w)
    % Resampling function based on weights
    index = find(rand <= cumsum(w), 1, 'first');
end
```

4.3 Αλγόριθμος tracking με φίλτρο Kalman

```
clear; close all; clc
%% Read video into MATLAB using VideoReader
video = VideoReader('sevenup.m4v');
nframes = video.NumFrames;
%% Calculate the background image by averaging the first 90 frames
temp = zeros(video.Height, video.Width, 3);
for i = 1:90
    frame = readFrame(video);
    temp = double(frame) + temp;
end
imbkg = temp / 90;
%% Reinitialize video reader to read frames from the start
video.CurrentTime = 0;
%% Initial position and size of the object
initial_xcenter = 181; % x-coordinate of the center of the object
initial_ycenter = 95; % y-coordinate of the center of the object
xwidth = 69;  % width of the bounding box
ywidth = 133;  % height of the bounding box
%% Calculate the initial corners based on the center and size
xcorner = initial_xcenter - xwidth / 2;
ycorner = initial_ycenter - ywidth / 2;
%% Initialization for Kalman Filtering
centroidx = zeros(nframes, 1);
centroidy = zeros(nframes, 1);
actual = zeros(nframes, 2); % Only need 2 states: position and velocity
%%  Kalman filter parameters for tracking in the x direction
% Time step (assuming frame rate is 1 frame per second)
dt = 1;
% Measurement noise covariance (variance of measurement noise)
R = 0.01;
% Process noise covariance (variance of process noise)
% Q controls how much we trust the model versus the measurements.
Q = [0.01, 0;
     0, 0.1];
% Initial error covariance matrix
P = [1, 0;
     0, 1];
% State transition matrix A for object moving only in the x direction
A = [1, dt;
     0, 1];
% Measurement matrix H, only measuring the x position
H = [1, 0];
```

```
%% Set the initial centroid
centroidx(1) = initial_xcenter;
centroidy(1) = initial_ycenter;
%% Initialize Kalman filter with the initial position
predicted = [centroidx(1); 0]; % [position; velocity]
%% Tracking loop
for i = 1:nframes
    frame = readFrame(video);
    imshow(frame);
    hold on;
    imcurrent = double(frame);
    %% Calculate the difference image to extract pixels with more than 80 (threshold)
change
    % Lower threshold: More sensitivity to changes, more noise.
    % Higher threshold: Less sensitivity to changes, risk of missing the object.
    threshold = 80;
    diffimg = (abs(imcurrent(:,:,1) - imbkg(:,:,1)) > threshold) ...
            | (abs(imcurrent(:,:,2) - imbkg(:,:,2)) > threshold) ...
            | (abs(imcurrent(:,:,3) - imbkg(:,:,3)) > threshold);
    %% Label the image and mark regions
    labelimg = bwlabel(diffimg, 4);
    markimg = regionprops(labelimg, 'BoundingBox', 'Centroid', 'Area');

    if isempty(markimg)
        % Use Kalman filter to predict if no object is detected
        predicted = A * predicted;
    else
        % Find the object closest to the predicted centroid
        distances = arrayfun(@(m) abs(m.Centroid(1) - predicted(1)), markimg);
        [~, idx] = min(distances);
        cc = markimg(idx).Centroid;

        centroidx(i) = cc(1);
        centroidy(i) = cc(2);

        % Convert to column vector for consistency
        measurement = [centroidx(i); 0];

        % Apply Kalman filter
        Ppre = A * P * A' + Q;
        K = Ppre * H' / (H * Ppre * H' + R);
        actual(i, :) = (predicted + K * (measurement(1) - H * predicted))';
        P = (eye(2) - K * H) * Ppre;

        % Update prediction
        predicted = A * actual(i, :)';
    end
    %% Plot the tracking rectangle after Kalman filtering
    rectangle('Position', [(predicted(1) - xwidth / 2), (initial_ycenter - ywidth /
2), xwidth, ywidth], 'EdgeColor', 'r', 'LineWidth', 1.5);
    plot(predicted(1), initial_ycenter, 'rx', 'LineWidth', 1.5);
    drawnow;
end
```