Θεωρία εκτίμησης και εφαρμογές

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Διατμηματικό ΠΜΣ στα Προηγμένα Συστήματα Υπολογιστών και Επικοινωνιών
Θεωρία εκτίμησης και εφαρμογές

Θεόδωρος Παναγιώτης Βαγενάς, 410
Πετρόπουλος Αλέξανδρος, 440

Θεσσαλονίκη Ιούνιος 2019

Εργασία 1η (φίλτρο Kalman):

Αρχικά στις παρακάτω εικόνες δίνονται οι εξισώσεις του μοντέλου κίνησης, του μοντέλου μέτρησης καθώς και αντίστοιχες Jacobian, που χρειάζονται για την υλοποίηση του φίλτρου Kalman.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \\ X_{1t} \\ Y_{1t} \\ X_{2t} \\ Y_{2t} \end{bmatrix} = \begin{bmatrix} 0.1v\cos(\theta_{t-1}) + x_{t-1} \\ 0.1v\sin(\theta_{t-1}) + y_{t-1} \\ \theta_{t-1} + 0.1w \\ X_{1t-1} \\ Y_{1t-1} \\ X_{2t-1} \\ Y_{2t-1} \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -0.1v\sin(\theta) & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.1v\cos(\theta) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Εικόνα: πίνακες μοντέλου κίνησης(αριστερά) και αντίστοιχη Jacobian(δεξιά)

$$\begin{bmatrix} d \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{(-X_{land} + x)^2 + (-Y_{land} + y)^2} \\ -\theta + \text{atan}_2(Y_{land} - y, X_{land} - x) \end{bmatrix} \begin{bmatrix} \frac{-X_{land}+x}{\sqrt{q}} & \frac{-Y_{land}+y}{\sqrt{q}} & 0 & \frac{X_{land}-x}{\sqrt{q}} & \frac{Y_{land}-y}{\sqrt{q}} \\ -\frac{-Y_{land}+y}{q} & -\frac{X_{land}-x}{q} & -1 & \frac{-Y_{land}+y}{q} & \frac{X_{land}-x}{q} \end{bmatrix}$$

$$where \;\; q = (-X_{land} + x)^2 + (-Y_{land} + y)^2$$

Εικόνα : συνάρτηση μοντέλου μέτρησης(αριστερά) και αντίστοιχος πίνακας Jacobian(δεξιά)

Για το μοντέλο μέτρησης έχουμε υλοποιήσει τον αλγόριθμο από το κεφ.10 του βιβλίου Probabilistic Robotics, όπου λαμβάνουμε υπόψιν τις μετρήσεις των εμποδίων μία κάθε φορά. Επίσης έχουμε τροποποιήσει το **predict()** , του αλγορίθμου του βιβλίου ώστε να εισάγουμε και κάποιο θόρυβο στα landmarks. Ο λόγος που το κάναμε αυτό είναι γιατί μπορεί ο αισθητήρας laser που χρησιμοποιούμε , να μην χτυπάει πάντα στο ίδιο σημείο του εμποδίου !

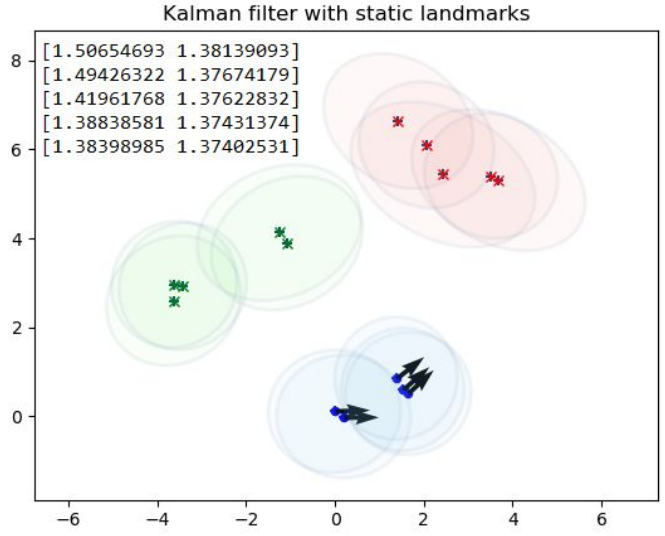
Επιπλέον υλοποιήσαμε και μία συνάρτηση **pi2pi** όπου καλείται κάθε φορά που γίνεται μία πράξη μεταξύ γωνιών (πρόσθεση, αφαίρεση). Ο λόγος που την υλοποιήσαμε είναι για να σιγουρευτούμε ότι όλες οι γωνίες θα βρίσκονται στο διάστημα [-pi,pi] . Σε διαφορετική περίπτωση , όταν κάναμε πράξεις μεταξύ των γωνιών θα είχαμε μεγάλα νούμερα και σαν συνέπεια θα εκτοξευόταν η τιμή του Kalman Gain, οπότε και θα είχαμε πολύ κακή εκτέλεση.

Στην συνέχεια δίνονται οι πίνακες με τις τιμές που επιλέξαμε μετά από πολλές δοκιμές που κάναμε :

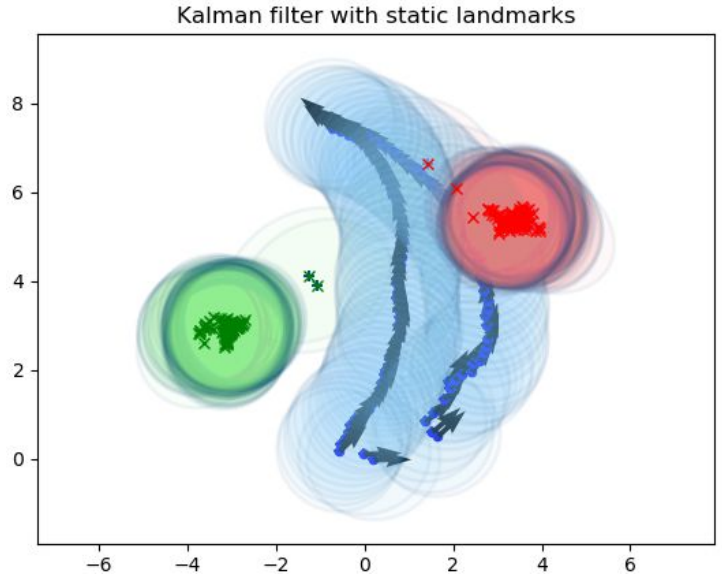
$$P = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.17 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad Q = \begin{bmatrix} 0.0001 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix} \quad R = \begin{bmatrix} 0.25 & 0.0 \\ 0.0 & 0.09 \end{bmatrix}$$

Πίνακες : P πίνακας αρχικής αβεβαιότητας , Q πίνακας θορύβου για το μοντέλο κίνησης , R πίνακας θορύβου για τις μετρήσεις

Σαν αρχικές τιμές για τα εμπόδια χρησιμοποιήσαμε την πρώτη μέτρηση από το radar ώστε να κάνουμε μια πρώτη εκτίμηση. Στην συνέχεια για την αρχική αβεβαιότητα (πίνακας P), δώσαμε λίγο πιο μεγάλες τιμές ώστε να φανεί η μείωση της αβεβαιότητας. Λόγω του ότι το πρόβλημα μας είναι πολύ απλό, το φίλτρο kalman συγκλίνει γρήγορα στις πρώτες κιόλας επαναλήψεις, οπότε και η αβεβαιότητα παραμένει σχετικά σταθερή.Το παραπάνω φαίνεται στην επόμενη εικόνα , όπου έχουμε τυπώσει τις 5 πρώτες επαναλήψεις μαζί με τις τιμές του μέτρου των ιδιοτιμών του πίνακα συσχέτισης :



Στην συνέχεια για τον πίνακα θορύβου του μοντέλου κίνησης επιλέξαμε την τιμή 0.01^2 , καθώς παρατηρήσαμε από το αρχείο control που μας δίνεται ότι η μέγιστη ταχύτητα είναι 2 , οπότε επί το dt, που είναι 0.1 , έχουμε ότι η μέγιστη μετατόπιση του ρομπότ θα είναι 0.2 m. Αν πάρουμε το 5-10% αυτής της τιμής προκύπτει και η τιμή του θορύβου μας. Παρατηρώντας και δοκιμάζοντας και άλλες τιμές είδαμε ότι είχαμε μια αρκετά καλή εκτέλεση με ομαλή κίνηση του ρομπότ και τα εμπόδια να κινούνται ελάχιστα γύρω από ένα σημείο. Στην συνέχεια δίνεται μια εικόνα τρεξίματος με τις παραπάνω τιμές:



Για την τελική θέση των landmarks κρατήσαμε την τελευταία θέσης τους, καθώς είδαμε ότι δεν είχαν μεγάλη απόκλιση.

*Στα αρχεία κώδικα (kalman_initial_version.py) παραδίδεται και η υλοποίηση του Extended Kalman στην απλή μορφή όπου λαμβάνονται μαζί όλες οι μετρήσεις (χωρίς εκμετάλλευση των αραιών πινάκων) για τα εμπόδια. Παρατηρήσαμε όμοια αποτελέσματα και στις δύο υλοποιήσεις.

Εργασία 2η (Particle filter με σταθερά εμπόδια):

Στη 2η εργασία δεχτήκαμε ως αρχικές θέσεις των εμποδίων αυτές που προέκυψαν από το Kalman και είναι οι εξής:

	x	**y**
1ο εμπόδιο	3.21846589	5.35298539
2ο εμπόδιο	-3.17810305	2.97454244

Αρχικοποιήσεις

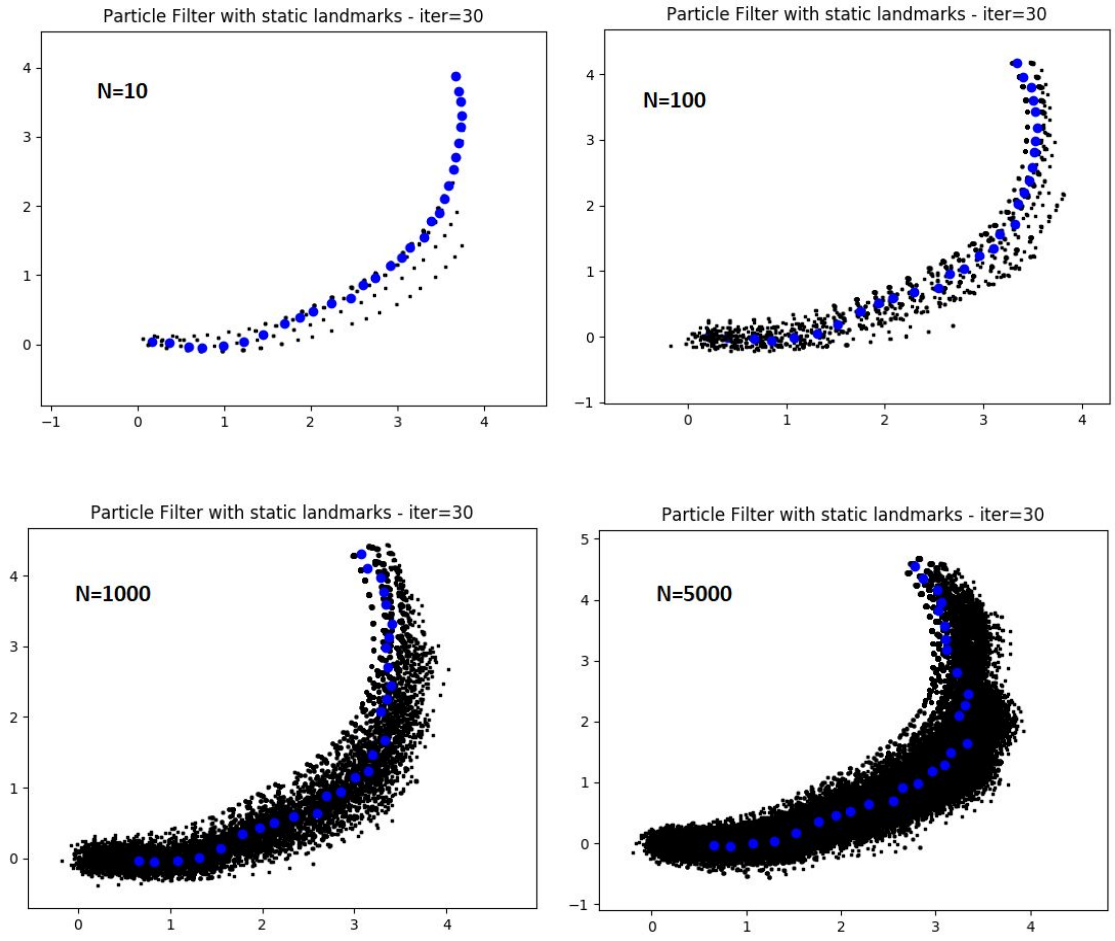
Ενώ για αρχική θέση και κατεύθυνση του ρομπότ θέσαμε (0,0,0).Στον αλγόριθμο του Particle Filter αρχικά πρέπει να ορίσουμε τι θα περιέχει κάθε particle. Επειδή λοιπόν από το πρόβλημα θέλουμε να υπολογίσουμε τη θέση του ρομποτ (x,y) και την κατεύθυνση θ το κάθε particle θα έχει μια τέτοια τριάδα (x,y,θ) που θα είναι μια πιθανή θέση του ρομποτ.
Στη συνέχεια πρέπει να αποφασιστεί πως θα γίνει η αρχικοποίηση των particles. Ελέγχθηκαν δύο τρόποι: με ομοιόμορφη κατανομή και με κανονική κατανομή. Στην περίπτωση της ομοιόμορφης κατανομής φτιάξαμε μια τέτοια κατανομή στα διαστήματα για χ: [-12,12] για y=[-12,12] για θ=[-π,π] και πήραμε particles από αυτήν. Στη 2η περίπτωση φτιάξαμε μια κανονική κατανομή με μέσο όρο την αρχική θέση που θέσαμε και σε κάθε στοιχείο προσθέταμε μία τιμή από κανονική κατανομή με την τυπική απόκλιση που ορίσαμε για τα std(x,y,θ) = [0.1,0.1,0.1] π.χ. particles[:, 0] = mean[0] + (randn(N) * std[0]) για το x. Προτιμήθηκε η κανονική κατανομή καθώς έβγαλε πιο σταθερά αποτελέσματα πράγμα που συμβαίνει γιατί αν θεωρήσουμε την αρχική θέση [0,0,0] ως 1η θέση το ρομπότ στη συνέχεια κινείται με μία ταχύτητα που θα βρίσκεται σε κάθε στιγμή t σε μία θέση γύρω στην θέση που βρισκόταν την χρονική στιγμή t-1 και όχι σε κάποια τυχαία θέση στο επίπεδο δεδομένου και ότι το dt=0.1 σχετικά μικρό. Η αρχικοποίηση των βαρών για κάθε particle δίνεται ως 1/N αφού αρχικά δίνουμε το ίδιο βάρος σε καθένα. Οι εξισώσεις για το τμήμα του predict() προέρχονται από το μοντέλο που δίνεται όμως σε κάθε μία εισάγεται και ένας μικρός θόρυβος με τον οποίο από τη μία μοντελοποιούμε το θόρυβο στα controls και το μοντέλο και από την άλλη με αυτόν τον τρόπο τα δείγματα που προέκυψαν από το resampling που είναι ίδια με άλλα διασκορπίζονται λίγο στο χώρο ώστε να γίνεται καλύτερη αναζήτηση των πιθανών λύσεων. Οι τυπικές αποκλίσεις των θορύβων έμειναν ίδιες με αυτές του Kalman. Οι εξισώσεις για κάθε particle φαίνονται παρακάτω:

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + 0.1\,(randn(N)std[0] + v)\cos(\theta_{t-1}) \\ y_{t-1} + 0.1\,(randn(N)std[1] + v)\sin(\theta_{t-1}) \\ 0.1randn(N)std[2] + \theta_{t-1} + 0.1w \end{bmatrix}$$

Επιλογή αριθμού Particles

Όσων αφορά την επιλογή του αριθμού των particles έγιναν δοκιμές για διαφορετικά νούμερα και πιο συγκεκριμένα για N=10,100,1000,5000 και επιλέχθηκε αυτή που κατά τη γνώμη μας έβγαζε σχετικά σταθερό αποτέλεσμα χωρίς να χρειάζεται πάρα πολύ συχνή δειγματοληψία

ενώ ταυτόχρονα παρουσίαζε σχετικά ομαλή πορεία οπότε χρησιμοποιήθηκε για N=1000. Με αυτό το N καλύπτεται ικανοποιητικά και με μεγάλη ακρίβεια η περιοχή γύρω από τη θέση του ρομπότ και άρα τα αποτελέσματα είναι πιο αξιόπιστα. Σε αυτήν την τιμή θεωρήθηκε ότι δεν καθυστερούσε υπερβολικά η εκτέλεση ενώ το αποτέλεσμα ήταν σχετικά σταθερό. Για μεγαλύτερο αριθμό όπως φαίνεται και στο διάγραμμα απλά αυξάνεται η πολυπλοκότητα και πολλά particles χρειάζεται γρήγορα να αντικατασταθούν μέσω του resample με αποτέλεσμα να μην κερδίζουμε κάτι. Παρακάτω βλέπουμε τις 30 πρώτες επαναλήψεις για N=10,100,1000,5000.



N	10	100	1000	5000
execution time(s)*	13.4	15.39	32.2	77.97

*Χρόνοι εκτέλεσης μαζί με την ταυτόχρονη ανανέωση του plot

Δημιουργία βήματος update- διόρθωσης

Στο τμήμα του αλγορίθμου όπου γίνεται το update πρέπει να επιλεχθεί ο τρόπος με τον οποίο θα αλλάζει το βάρος που δίνεται σε κάθε particle. Αυτό γίνεται με τον έλεγχο του κατά πόσο ταιριάζει κάθε particle με τις μετρήσεις που παίρνουμε. Αρχικά δοκιμάστηκε να αξιοποιηθεί μόνο η διόρθωση από την απόσταση του ραντάρ αλλά προκειμένου να γίνουν πιο σταθερά και ακριβή τα αποτελέσματα αξιοποιήθηκε και η γωνία. Σε αυτήν για να λάβουμε υπόψιν κάθε μέτρηση για την απόσταση και την γωνία τη σχετική με τα εμπόδια από το radar χρησιμοποιούμε μία πολυμεταβλητή κανονική κατανομή (multivariate normal distribution) ως εξής: Υπολογίζεται η διαφορά της απόστασης που υπολογίζεται από το particle προς τη θέση του εμποδίου από την απόσταση πού δείχνει το radar. Επίσης ως $2^{η}$ μεταβλητή υπολογίζεται η διαφορά της γωνίας που υπολογίζεται από το μοντέλο μέσω του particle και του εμποδίου και της γωνίας που δίνεται από το ρανταρ. Αυτές οι διαφορές χρησιμοποιούνται ως οι μέσες τιμές στην κανονική κατανομή (2 μεταβλητών) με τυπική

απόκλιση (std) ίσες με αυτές που δίνονται για το ραντάρ. Σε αυτήν υπολογίζονται οι τιμές για εισόδους τις μετρήσεις του ραντάρ. Έτσι έχουμε μια ποσότητα που δείχνει πόσο ταιριάζει το κάθε particle με την μέτρηση και υπολογίζονται τα βάρη. Για τη πολυμεταβλητή κανονική κατανομή υλοποιήθηκε η συνάρτηση normpdf(x, mu, std) . Μετά από κάθε τέτοιο υπολογισμό γίνεται κανονικοποίηση των βαρών. Παρακάτω παρουσιάζεται ο κώδικας για το update και ο τύπος της συνάρτησης normpdf.

```
def update(self,z,z_phi):
    for i in range(0,self.landmarks.shape[0]) :
        current_lm = self.landmarks[i]
        distance = np.linalg.norm(self.particles[:, 0:2] - current_lm, axis=1)
        temp = calc_angle(np.arctan2(current_lm[1]-self.particles[:,1],current_lm[0]-self.particles[:,0])-self.particles[:,2])
        dist2 = np.array(temp-z_phi[i])
        dist2= calc_angle(dist2)
        full_dist = (np.array([distance,dist2])).T
        w1 = [normpdf(np.array([z[i],z_phi[i]]),full_dist[ttt],self.R_std) for ttt in range(0,full_dist.shape[0])]
        self.weights *= w1

    self.weights = self.weights + 10**(-10)

    self.weights = self.weights / sum(self.weights)
```

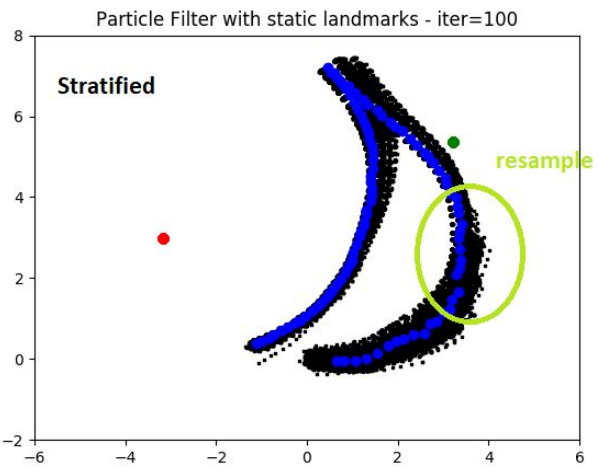
$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Τεχνικές επαναδειγματοληψίας (Resampling)

Μεγάλη επίδραση στο αποτέλεσμα είχε η τεχνική resampling που θα χρησιμοποιηθεί. Πρέπει πρώτα να οριστεί το πότε θα γίνεται το resampling και για αυτό χρησιμοποιήθηκε το effective N που μετράει πόσα particles συνεισφέρουν ουσιαστικά στην κατανομή και ορίστηκε ως κατώφλι στο neffective (3N/4)=75% με neff= 1/sum(weights^2). Για n effective δοκιμάστηκαν διάφορες τιμές στο βαθμό εκτός του τετραγώνου κάνοντας το resampling πιο επιθετικό ή πιο χαλαρό αλλά δεν επετεύχθει ουσιαστική βελτίωση με αποτέλεσμα να διατηρηθεί το αρχικό.
Δοκιμάστηκαν οι παρακάτω μέθοδοι resampling για την αντικατάσταση των particles που δεν συνεισφέρουν ουσιαστικά με κάποια τα οποία έχουν μεγαλύτερο βάρος.

Απλό resample (multinomial) στο οποίο υπολογίζεται το αθροιστικό άθροισμα (cumulative sum) των βαρών και κάποιοι τυχαίοι αριθμοί (ομοιόμορφα) στο διάστημα [0,1] και με δυαδική αναζήτηση βρίσκεται το βάρος που είναι πιο κοντά στον αριθμό. Έτσι οι περιοχές με μεγαλύτερο βάρος έχουν μεγαλύτερη πιθανότητα να δώσουν νέο particle.
Residual Resampling στο οποίο πολλαπλασιάζουμε τα κανονικοποιημένα βάρη με το N οπότε το ακέραιο μέρος δείχνει πόσα δείγματα θα πάρουμε από κάθε particle. Και στη συνέχεια υπολογίζονται τα υπόλοιπα (δηλαδή αν βάρος*N=7,8 τότε 7,8-7=0.8) για να συμπληρωθούν και όσα praticles από αυτά που θα αντικατασταθούν υπολείπονται. Γίνεται με τον προηγούμενο τρόπο δειγματοληψίας και άρα πιο μεγάλο βάρος σημαίνει ότι μπορούν πιο πιθανά να ξαναεπιλεχθούν.

Stratified Resampling: Αυτό είναι που προτιμήθηκε και κατά τις τελικές εκτελέσεις. Σε αυτό υπολογίζεται και πάλι το cumulative sum αλλά αυτή την φορά διαιρείται ο χώρος σε ίσα τμήματα (N) και έτσι επιλέγονται τα δείγματα με βάση το cumulative sum αλλά τυχαία σε κάθε τμήμα. Με αυτό γίνονται πιο ομοιόμορφες οι επιλογές.

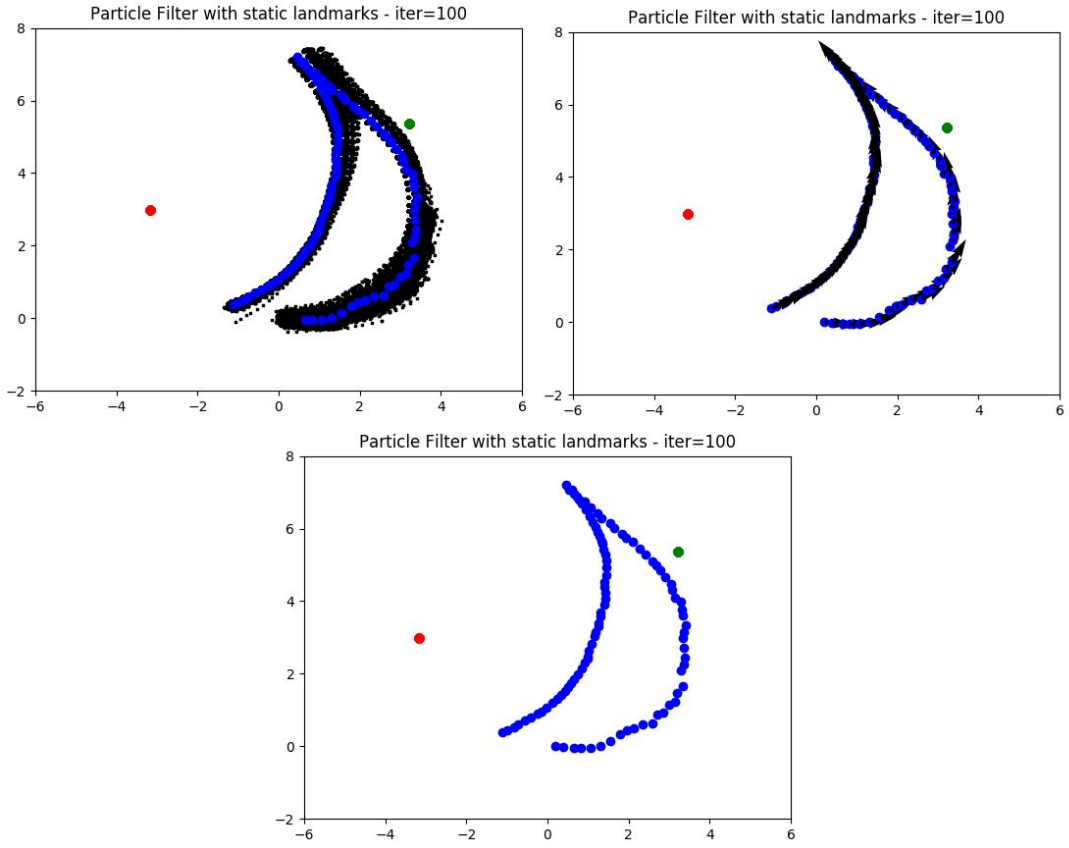


Systematic resample: χωρίζεται και πάλι ο χώρος σε N τμήματα όμως ταυτόχρονα υπολογίζεται ένας τυχαίος αριθμός ο οποίος χρησιμοποιείται ως η θέση μέσα σε κάθε

τμήμα από τα N από όπου θα προκύψει το δείγμα.

*Αξίζει να σημειωθεί ότι όπου χρησιμοποιήθηκαν γωνίες έγινε διόρθωση τους όπως και στο Kalman.

Διαγράμματα με particles, με γωνία θ, μόνο με θέσεις



Τελική θέση ρομπότ [x,y,θ] = [-1.10460698 0.37738887 0.42084822]. Τέλος, παρατηρούμε ότι υπάρχει μεγάλη ομοιότητα μεταξύ των αποτελεσμάτων του Kalman και του Particle FIlter.

Εργασία 3η (Particle filter με κινούμενο εμπόδιο):

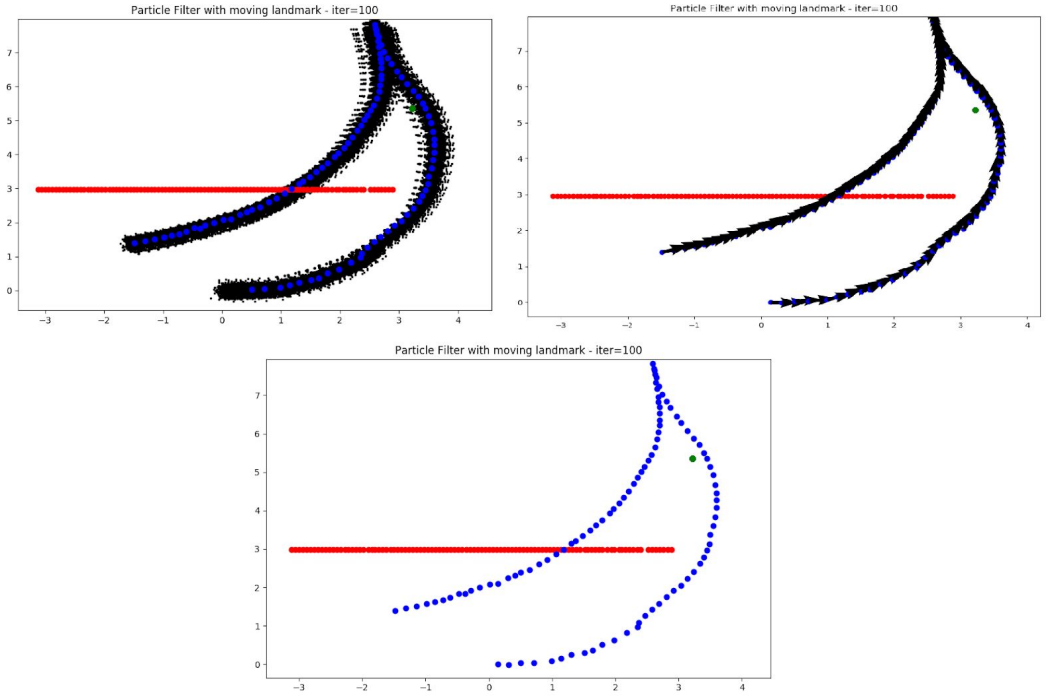
Στην 3η εργασία πρέπει να εκτιμηθούν εκτός από τη θέση και τη κατεύθυνση του ρομπότ (x,y,θ) και η ταχύτητα και η θέση ως προς τα x του landmark που κινείται. Επομένως το νέο διάνυσμα για κάθε particle θα έχει τη μορφή: [x,y,θ,xob,vob] . Επειδή πρόκειται για σταθερή ταχύτητα η θέση δίνεται από τον τύπο xob=v*dt .

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \\ X_t^{(lan2)} \\ ut_t^{(land2)} \end{bmatrix} = \begin{bmatrix} x_{t-1} + 0.1\,(randn(N)std[0] + v)\cos(\theta_{t-1}) \\ y_{t-1} + 0.1\,(randn(N)std[1] + v)\sin(\theta_{t-1}) \\ 0.1randn(N)std[2] + \theta_{t-1} + 0.1w \\ X_{t-1}^{(land2)} + 0.1std[3] + 0.1u_{t-1}^{(land2)} \\ std[4] + u_{t-1}^{(land2)} \end{bmatrix}$$

Εδώ η αρχικοποίηση με κανονική κατανομή για τα particles έγινε με τυπικές αποκλίσεις [0.1,0.1,0.01,0.01,0.1] αντίστοιχα. Τα βάρη αρχικοποίηθηκαν όμοια ως 1/N . Το βήμα του update και του resample έγινε όπως και στην εργασία 2. Για N για τους ίδιους λόγους επιλέχθηκε το 1000 γιατί είναι αρκετό παρόλου που αυξήθηκε η πολυπλοκότητα του της εκτίμησης με την αύξηση των καταστάσεων.

Για την επιλογή της ταχύτητας του landmark το οποίο μπορεί να πηγαίνει είτε δεξιά είτε αριστερά καθώς δεν διευκρινίζεται από την εκφώνηση διερευνήθηκαν οι τιμές του ραντάρ και έχοντας προσπαθήσει να διατηρήσουμε τα βήματα στα οποία μειώνεται η ταχύτητα και στη συνέχεια αυξάνεται αποφασίστηκε ότι το εμπόδιο πιθανότατα πάει προς τα δεξιά και με ταχύτητες 0.6-0.7 ώστε να προκύπτει το σχήμα που φαίνεται παρακάτω και συμφωνεί με τις αυξομειώσεις των αποστάσεων και γωνιών του ραντάρ. Με 0.7 τελικές θέσεις particle [x,y,θ,xob,vob]=[-1.48566345 1.40265512 0.26497524 3.88522551 0.70682629].
Με 0.6 τελικές θέσεις particle [x,y,θ,xob,vob]=[-1.48566345 1.40265512 0.26497524 2.88522551 0.60682629].

Διαγράμματα με κινούμενο εμπόδιο (vob=0.6)



Αναφορές :

http://sait.cie.put.poznan.pl/38/SAIT_38_02.pdf
https://link.springer.com/article/10.1631/FITEE.1500199
https://filterpy.readthedocs.io/en/latest/
http://www.probabilistic-robotics.org/