Aarhus Tech
SRP

22. december 2017

Monte Carlo Lokalisering

Forfatter
Jacob Emil Ulvedal Rosborg

Vejleder
Mikkel Stouby Petersen
Jørn Sanggaard

Abstract

In the following, you will read and learn about Monte Carlo Method and and how it is to be perceived as a general thought of mind that have evolved into different methods and algorithms of approximating problems. It was first developed by Stanislaw Ulam during the Manhattan Project and was used to solve the problem of neutron diffusion. The problem of neutron diffusion is complex, so much in fact that they could not derive it algebraically. They therefor looked into means of approximating neutrons diffusion and that let to Monte Carlo Method. Further more, you will read how this method was adopted to solve integrates that is not other wise solvable. The methods for solving such integrates is often referred to as numerical integration and in the following we will be using the method Hit-or-Miss to approximate π. At last you will read how the method can be used to locate an autonomous robot in given localized area. The method of localizing is not inherently robust, but can be made so by introducing noice particles.

Indhold

1 Indledning

Nogle problemer er så komplicerede, at de næsten er umulige at analysere, modellere og løse algebraisk. For eksempel den kaotiske og tilfældige proces, der finder sted når man spalter uran-235 (neutron diffusion). Dette problem stod forskerne overfor i 1940erne under udviklingen af atombomben i Manhattan projektet. Her blev Monte Carlo Simulering anvendt til at simulere neutroners vandring. Det blev brugt til at vurdere hvor langt neutroner fra diverse henfald ville bevæge sig i gennem forskellige materialer. Denne viden blev brugt til at designe reaktorer med tilstrækkelig beskyttelse [3].

Begrebet Monte Carlo Metod*en* dækker over en række metoder, der kan bruges til at analyse problemer, som ikke fremstår løsbare, såsom *neutron diffusion*. For eksempel findes der metoder, som Monte Carlo Simulering og Monte Carlo Lokalisering. Variationer af disse metoder findes også indenfor Finans og Medicin.
Denne opgave vil fokusere på numerisk integration ved brug af Monte Carlo Metoden Hit-or-Miss og ydermere vil opgaven komme ind på lokalisering af en autonom robot ved brug af Monte Carlo Lokalisering. Derudover vil opgaven også redegøre for Stokastiske Variabler, og de egenskaber der er nødvendige for at forklare Monte Carlo metoderne.

Monte Carlo Metoden blev først rigtigt anvendt da computeren blev så teknologisk udviklet, at den kunne udføre disse simuleringer, også kaldet eksperimenter, for os. Disse simuleringer kræver et forholdsvis stort antal gentagelser, et antal der både kræver arbejdskraft og tid af umådelige proportioner. Alt dette for at opnå et resultat, der ikke altid vil være brugbart. Dette resultat vil være en approximering hvorimod en algebraisk tilgang vil udlede værdien og derved være eksakt. Til gengæld er man med Monte Carlo Metoden i stand til at tackle problemer der så komplekse i sin natur, at det ikke er praktisk muligt at udlede disse problemer algebraisk [4].

I nedenstående fremgår denne SRP opgaves opgaveformulering. Det er netop denne, som vil blive besvaret løbende gennem opgaven. SRP opgavens skrives med udgangspunkt i fagene Matematik A og Robotteknik A.

1.1 Opgaveformulering

(I) Redegør for, hvad Monte Carlo-algoritmer er, og giv eksempler både praktiske og teoretiske anvendelser. For eksempel i forbindelse med numerisk integration.

(II) Forklar centrale egenskaber ved stokastiske variable i det omfang det er nødvendigt for at forstå algoritmernes virkemåde.

(III) Vis, hvordan Monte Carlo-algoritmen kan anvendes til lokalisering af robotter. Kom herunder ind på, hvordan algoritmen kan implementeres.

(IV) Diskuter Monte Carlo-metodens muligheder og begrænsninger i forbindelse med anvendelse i en konkret autonom robot.

2 Stokastiske variabler

En væsentlig del af Monte Carlo Metodens teori og praksis bygger på stokastiske variabler. For at forstå stokastiske variabler kræves der en forståelse for simplere begreber indenfor statistik. Disse begreber er: udfaldsrum, delmængde, sandsynlighedsfunktion samt sandsynlighedsfelt. Disse begreber redegøres der for i nedenstående afsnit.

2.1 Udfaldsrum og delmængder

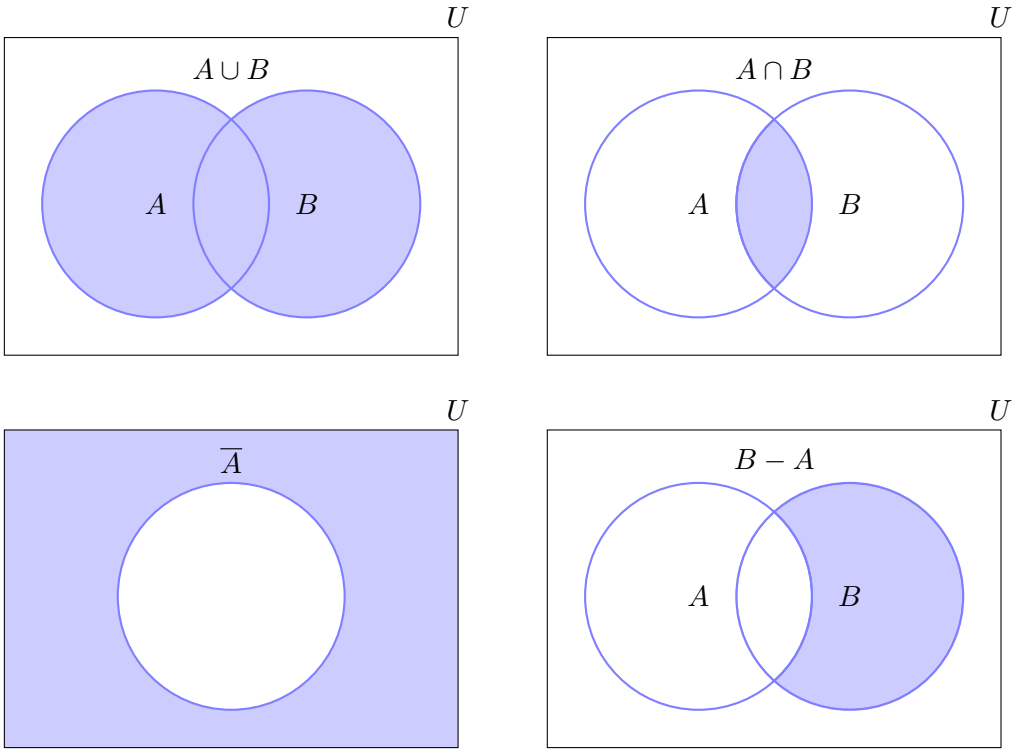
Der findes to typer af udfaldsrum, et diskret og kontinuert udfaldsrum. Forskellen på disse to er, at der i et diskret udfaldsrum kun er et givent antal mulige udfald. Et eksempel på et diskret udfaldsrum er en sekssidet terning. Denne sekssidet terning vil have et udfaldsrum U, der indeholder elementerne $1, 2, 3, 4, 5, 6$. Det kan skrives som $U = \{1, 2, 3, 4, 5, 6\}$. Altså er der tale om et udfaldsrum med seks elementer, hvilket skrives $|U| = 6$ eller som n. Et kontinuert udfaldsrum kunne eksempelvis være højden af elever i en klasse. I dette tilfælde kan udfaldsrummet være defineret som $U = [150cm; 210cm]$. Udfaldsrummet er kontinuert da elevers højde kan variere med et infinitesimal. En elev a kan være en uendelig lille stykke højere end elev b. Et sådan udfaldsrum indenholder ikke et endeligt antal elementer [8].

En delmængde er en mængde af et udfaldsrum. Delmængden beskrives ofte med A eller B. Hvis vi endnu engang tager udgangspunkt i den sekssidet terning, kan en delmængde af ternings udfaldsrum være $A = \{1, 3\}$. En delmængde indeholder derved en eller flere elementer fra udfaldsrummet.

Ydermere vil et bestemt element i et udfaldsrum eller delmængde denoteres u_i eller a_i, altså for henholdsvis A og U. Det vil sige, at hvis vi har en mængde Q vil et bestemt element i Q denoteres som q_i. Desuden findes der en række operatorer, der beskriver delmængders relationer til hinanden samt udfaldsrummet.

1. $A \cup B$: udfaldet ligger i enten A eller B, evt. i både A og B
2. $A \cap B$: udfaldet ligger i både A og B
3. $A \backslash B$: udfaldet ligger i A og ikke B
4. A^c: udfaldet ligger ikke i A (dette kan også denoteres $\bar{A}$)

Betegnelserne er de samme, som vi bruger i mængdelæren, og vi taler derfor om foreningsmængden $A \cup B$, fællesmængden $A \cap B$, mængdedifferensen $A \backslash B$ samt komplementærmængden A^c [8, p. 16]. Nedenstående figurer, også kaldet venn-diagrammer, illustrerer overstående mængder og deres operatorer.



2.2 Sandsynlighedsfunktion

Sandsynlighedsfunktion denoteres P og beskriver sandsynligheden for et element i udfaldsrummet U. For eksempel kunne $P(\{1,2\}) = 0{,}8$ og $P(3) = 0{,}2$, hvor udfaldsrummet $U = \{1,2,3\}$. Dette betyder, at sandsynligheden for 1 eller 2 er 80%, hvorimod sandsynligheden for 3 er 20%. Desuden kan vi bruge additionsloven til at udlede den totale sandsynlighed for udfaldsrummet [8].

$$P(1) + P(2) + P(3) = 1 \tag{2.1}$$
$$P(\{1,2\}) + P(3) = 1 \tag{2.2}$$

Desuden kan $\{1,2\}$ også beskrives som en delmængde af U på formen $A = \{1,2\}$. Herved gælder det, at $P(A) = P(1) + P(2)$.

Sandsynlighedsfunktion kan også være afhængig af en anden sandsynlighed. Et eksempel på dette kunne være en kasse is. Kassen med is indeholder fire slags is, hvor 50% af isen er med mørk chokolade og de resterende 50% er med hvid chokolade. Lad denne sandsynlighed værende betegnet med P_c. Desuden har 70% af de is der har mørk chokolade, mandler og 50% af isene med hvide chokolade også mandler. Denne vil således være beskrevet som $P(u_i|P_c)$. Herved er sandsynligheden for udfaldet u_i afhængig af P_c.

2.3 Sandsynlighedsfelt

Et sandsynlighedsfelt findes på to former. Det endelige - også kaldet det diskrete sandsynlighedsfelt - samt det kontinuærte sandsynlighedsfelt også kaldet et ikke-endeligt sandsynlighedsfelt. Et sandsynlighedsfelt kan denoteres (U, P) og er bestående af et udfaldsrum

U og en sandsynlighedsfunktion P. Hvis der er tale om et symmetrisk sandsynlighedsfelt, vil følgende være gældende for P.

$$P(U) = 1 \tag{2.3}$$

$$P(u_1) + P(u_i) + \cdots + P(u_n) = 1 \tag{2.4}$$

$$P(u_1) = P(u_i) = \cdots = P(u_n) = \frac{1}{|U|} \tag{2.5}$$

De to udsagn udtrykker derved at sandsynligheden for $P(U_i)$ er i intervallet $[0; 1]$ samt at den samlede sandsynlighed for feltet er 1, hvor $1 = 100\%$. I et ikke-symmetrisk sandsynlighedsfelt vil de to første regler af de overstående tre gælde, men sandsynligheden for de enkelte elementer i udfaldsrummet er ikke nødvendigvis lig hinanden. Altså kan $P(u_1) \neq P(u_2)$ [8].

2.4 Stokastiske variabler

Der findes to typer af stokastiske variabler. Disse er begge denoteres med X og desuden er x_i et konkret udfald. Disse to typer er henholdsvis diskret og kontinuert. En stokastiske variable er en variable, som kan tage alle værdier i et givent udfaldsrum med sandsynligheden P, altså $P(X = u_i) = P(u_i)$. Forskellen mellem diskrete og kontinuerte stokastiske variabler er udfaldsrummet og de tilhørende regneregler. Et eksempel på en stokastiske variabel er Europæisk Roulette. Dette spil har et symmetrisk sandsynlighedsfelt med udfaldsrummet $U = \{0, 1, \ldots, 36\}$, hvilket indeholder $37 = |U|$ udfald. Det vil altså sige, at den kugle man smider ned i spillet kan beskrives som en stokastiske variabel X, og X har en lige stor sandsynlighed for at blive et element i udfaldsrummet, da der er tale om et symmetrisk sandsynlighedsfelt, hvilket betyder at $P(u_1) = P(u_i) = \cdots = P(u_n)$, det betyder at $P(X = 1) = P(X = 4)$. Ydermere er roulettens udfaldsrum opdelt i tre farver; grøn, rød og sort. Disse kan betegnes som delmængderne G, R og S. Vi kan beskrive sandsynligheden for, at den stokastiske variabel X ville være et element i delmængden således $P(X \in S) = P(X \in R) = \frac{18}{37}$ og $P(X \in G) = \frac{1}{37}$. Delmængden G kunne også have været defineret som $G = (R \cup S)^c$, altså de udfald som ikke falder ind under hverken R eller S, eller $G = 0$ [1]

Lad desuden $\mathbb{E}[X]$ være den forventede værdi af X. Den forventede værdi skal forstås næsten som et gennemsnit. I det ovenstående eksempel vil den forventede værdi X være $\mathbb{E}[X] = 18$. Det kan udregnes på følgende måde:

$$\mathbb{E}[X] = \sum_{i=1}^{n} \frac{u_i}{n} \tag{2.6}$$

Dette er kun tilfældet ved et symmetrisk sandsynlighedsfelt, hvis sandsynligheden for de forskellige udfald er forskellige fra hinanden, altså $P(u_1) \neq P(U_2)$. Her vil man kunne anvende følgende formel for den forventede værdi af X.

$$\mathbb{E}[X] = \sum_{i=1}^{n} u_i \times P(u_i) \tag{2.7}$$

Stokastiske variabler på en computer er ofte repræsenteret som et pseudo-genereret tilfældigt tal, hvilket er et tilfældigt tal udvalgt af en computer. Disse tilfældigt genererede

tal bliver brugt til at udføre eksperimenter og simulering af Monte Carlo Metoden [4]. Dette vil komme til udtryk i det følgende afsnit hvori det vil blive vist, hvordan Monte Carlo Simulering gør brug af Stokastiske variabler.

3 Monte Carlo Metoden

I dette afsnit vil opgaven vise og redegøre for hvad Monte Carlo Metoden er, og hvordan den kan anvendes til at udregne et integral.

Monte Carlo Metoden har sit navn efter Stanislaw Ulams onkel, som ofte spillede på Casino Monte Carlo i Monaco. Metoden refererer herved til de sandsynligheder og de tilfældige udkom der kan finde sted i sådanne spil. Ulam var en af de forskere der arbejdede på Monte Carlo Metoden under Manhatten Projektet, som blev iværksat i 1942 [2]. Metoden blev udviklet for at kunne simulere neutroners diffusion. Det var nødvendigt at udvikle og benytte denne metode, da problemet var for komplekst til at kunne blive afledt algebragisk eller blive løst med datidens metoder. Metoden blev først anvendt på gamle analoge computere, hvilket begrænsede kompleksiteten af simulationen [3].

Der ud over er Monte Carlo Metoden også en algoritme, hvilket betyder at det er en process som er udført i nogle logiske trin. Trinene beskriver udførelse af metoden. En algoritme kunne for eksempel være de trin, som er involveret i sorteringen et sæt kort. Lad kortspillet være et sæt kort og et element værende ét enkelt spillekort. Kortspillet kan således sorteres på følgende måde:

1. Vælg et vilkårligt element e_i fra sættet.
2. Placere elementet bagerst i sættet.
3. Sammenlign nu e_i med elementet lige før e_{i-1} hvis $e_i > e_{i-1}$ bytter de plads. Dette gentages indtil $e_i < e_{i-1}$ eller $i = 0$.
4. Gentag denne process indtil kortspillet er sorteret.

Denne sorterings algoritme er også kaldet boblesortering og navnet kommer af måden elementerne i sættet bobler til toppen. [7]

I det følgende afsnit vil opgaven give et eksempel på, hvordan spillet Plat eller Krone kan ses som en simpel Monte Carlo Simulering.

3.1 Plat eller Krone

Et eksempel på udførelse af Monte Carlo Simulering er spillet Plat eller Krone. Spillets regler er som følgende:

1. Hver spiller satser på en side af mønten.
2. Mønten kastes op i luften.
3. Vinderen er den spiller hvis sats er den side af mønten der vender op af.

Man kan også beskrive Plat og Krone med et symmetrisk sandsynlighedsfelt (U, P), hvor udfaldsrummet $U = \{Plat, Krone\}$ og sandsynlighedsfunktionen P, hvor følgende er sandt, da det er et symmetrisk sandsynlighedsfelt:

$$P(Plat) = P(Krone) = \frac{1}{2} \tag{3.1}$$

Vi kan desuden beskrive mønten med en stokastiske variable X. Den forventede værdi af X kan ligedes beskrives som værende $\mathbb{E}[X] = \frac{1+2}{2} = 1{,}5$ hvis $Plat = 1$ og $Krone = 2$ altså den gennemsnitlig værdi af udfaldsrummet.

Man kan finde frem til en approximering af den forventede værdi af X ved brug af Monte Carlo Simulering. I den sammenhæng lad $\mathbb{E}[X]$ være den forventede værdi af X. Vi kan derved opskrive følgende udsagn, hvor x_i er en konkret simulering af X:

$$\mathbb{E}[X] = \sum_{i=1}^{n} \frac{x_i}{n} \tag{3.2}$$

Vi kan udføre en simulering der simulere 1000 eksperimenter, hvor X er uafhængig.

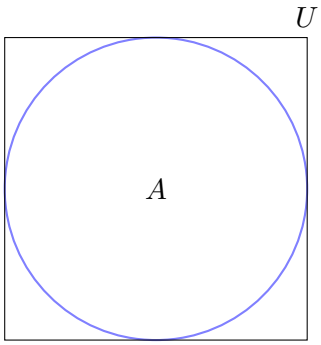
```
import random

random.seed(42)

print(sum([random.randint(1,2) for _ in range(1000)]) / 1000.000)
```

Dette giver os $\mathbb{E}[X] = 1{,}519$. Hvis vi udfører dette eksperiment 100 gange vil resultat varierer. Det skyldes, at X er tilfældigt bestemt for hver simulering og er uafhængig af forgående simuleringer. Det betragtes som normal opførelse for en fair mønt [1].

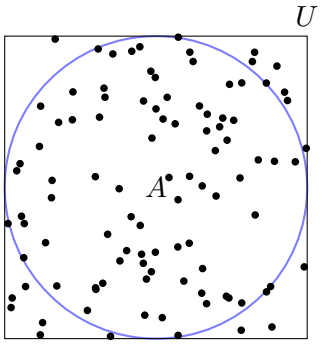
3.2 Numerisk integration

Det overstående eksempel med Plat eller Krone virker måske lidt dumt da vi med nemhed kan udlede det algebraisk ved at tage summen af udfaldsrummet og dele det med antallet af mulige udfald. Et andet tilfælde, hvor man kan anvende Monte Carlo Metoden er numerisk integration. I nogle tilfælde er det let at udlede integralet algebragisk, men der findes også tilfæde hvor det er umuligt. Disse figurer eller funktioners integraler kan udledes ved hjælp af numeriskemetoder, heriblandt Monte Carlo Metoden også kaldet Hit-or-Miss. [4]

Lad os betragte enhedscirkelen, der befinder sig inden for et kvadrat med dimensionerne 2×2. Denne figurs integral kan let udledes, men er valgt for nemheds skyld.



Vi kan starte med at bestemme udfaldsrummet af den overstående figur. Udfaldsrummet er U og indenholder alle de punkter der findes i kvadratets plan. Det integral vi ønsker at finde er cirklen, hvor vi lader de punkter der befinder si i cirklen være delmængde A. Desuden er sandsynligheden P for alle udfald i udfaldsrummet U lig hinanden. Det vil sige, at to givne udfald i U er sandsynligheden $P(u_1) = P(u_2)$. Dette betyder at der er tale om et symmetrisk sandsynlighedsfelt, med et udfaldsrum der er bestående af punkter der findes på kvadratets plan. Den stokastiske variabel X vil ligeledes repræsentere et muligt punkt på kvadratets plan. Dette felt er i teorien et kontinuert sandsynlighedsfelt, hvilket også ville gøre vores stokastiske variabel til en kontinuerlig variabel.
Hvis vi har n stokastiske variabler X og n' betegner de stokastiske variabler, der ligger inden for cirklens areal altså således at hvis $X = (0{,}5; 0{,}5)$ vil det være et udfald i delmængden A. Det betyder, at alle elementer i delmængden A vil opfylde $a_i x^2 + a_i y^2 \leq 1^2$, hvor a_i er et konkret tilfælde og $a_i x$ er dets x koordinat samt $a_i y$ er dets y koordinat. Nedenstående figur viser et eksempel på n antal konkrete stokastiske variabler x_i i udfaldsrummet U, hvor n' er de konkrete x_i som befinder sig i A.



Vi kan herved udregne en approximering af sandsynligheden for $P(A)$.

$$P(A) = \frac{n'}{n} \tag{3.3}$$

Hvis vi ønsker af finde arealet af cirklen skal vi gange med arealet af vores udfaldsrum, da dette var et kvadrat med dimensionerne 2×2 får vi følgende udtryk.

$$Areal(A) = \frac{n'}{n} \times 4 \tag{3.4}$$

En simulering af dette kan udtrykkes i følgende Python kode, hvor vi har 100000 konkrete stokastiske variabler

```
import random

random.seed(42)

kvadrattet = [(random.uniform(-1,1),
               random.uniform(-1,1)) for _ in range(100000)]
cirklen = [p for p in kvadrattet
             if p[0]*p[0] + p[1]*p[1] <= 1]

nk = len(kvadrattet) # lad dette vaere n
nc = len(cirklen)    # lad dette vaere n'
a = 2 * 2            # lad dette vaere arealet af kvadrattet

print("areal = {:d}/{:d}*{:d}={:f}".format(nc, nk, a, nc/nk*a))
```

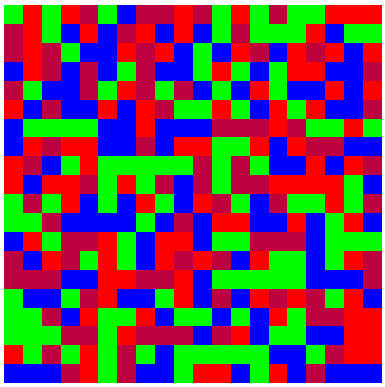
Ud fra ovenstående simulering, vil arealet af cirklen være 3,140280 og derved vil $\pi = 3{,}140280$. Da fejlen ved udregningen af et integral på denne måde er $\sqrt{\frac{1}{n}}$ ville vi finde at vi skulle øge antallet af punkter med en faktor 100 for at reducere fejlen med en faktor 10 [4].

4 Monte Carlo Lokalisering

I dette afsnit vil opgaven vise, hvordan Monte Carlo Lokalisering kan anvendes til at lokalisere en konkrete autonom robot i et kendt område.

Der tages udgangspunkt i en autonom robot, der har fire hjul og mindst to elektriske motorer med enkoder. Elektriske motorer med enkoder afgiver et elektrisk signal n antal gange per total rotation af motorens akse. Dette kan eksempelvis være implementeret ved brug af en hall sensor, der måler på en magnet eller en optisk sensor, som måler på et optisk gitter der følger aksen. Således er robotten i stand til at udregne, hvor langt den har bevæget sig i en given retning. Denne robot kunne f.eks. være en reklamerobot, som kører rundt i et givent område og har en bakke med chokolade som forbipasserende kan tage af. Dette område kunne være en banegård eller en butik. Robotten har til dette formål en sensor, der er placeret på undersiden. Sensoren måler farven af det underlag som robotten bevæger sig på. Sensoren måler et resultat på formen (r, g, b), hvor $(0, 0, 0)$ eksempelvis svarer til farven sort og $(255, 0, 0)$ svarer til farven rød [5].

Med udgangspunk i overstående robot, der implementerer et Dead-Reckoning algoritme, kan robotten lokaliseres på et kendt område ved brug af Monte Carlo Lokalisering. Det kendte område kan betragtes som et kort. I dette konkrete eksempel er kortet kvadratisk, 20×20 meter, og er bestående af fliser med forskellige farver. Disse farver er henholdsvis rød, grøn, blå og lilla, som illustreret på nedenstående figur.



Altså vi har et kendt kort, hvor der er 400 fliser af forskelige farver, hvilke også fungerer som punkter. Punkterne er repræsenteret som elementer i udfaldsrum U. Udfaldsrummets elementer er punkter på formen $u_i = (x, y)$ og $rgb(u_i)$ er flisens farven på formen RGB. Hvis vi har en rød flise på punktet $(1, 2)$ vil $rgb((1, 2)) = (255, 0, 0)$. Dette udfaldsrum er en del af et sandsynlighedsfelt, hvilket betyder at robotten har en sandsynlighed for at befinde sig på et givent punkt. Ud fra dette sandsynlighedsfelt kan vi definere en stokastisk variabel X, som kan være hvilket som helst punkt i udfaldsrummet U, hvor x_i er et konkrete punkt.

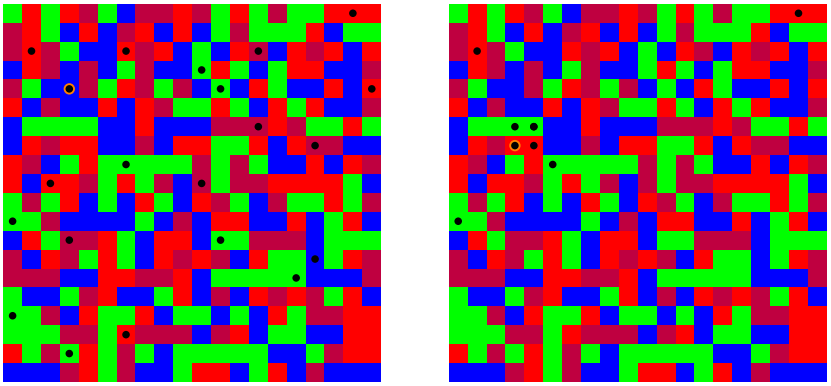
Algoritmen består i et løbende antal iterationer, hvor under hver iteration bliver der simuleret et n antal konkrete stokastisk variabler x_i. Disse konkrete punkter bliver også kaldet for partikler, hvilket bliver tydeligt på de kommende illustrationer. Desuden er

sandsynlighedsfunktionen P afhængig af om der er tale om den første iteration af algoritmen eller om det er løbende iterationer. For første iteration er sandsynlighedsfunktionen den samme som ved et symmetrisk sandsynlighedsfelt. Der er her tale om en jævn fordeling af partiklerne, men de efterfølgende iterationer er afhængig af kun og kun af den forgående iterations partikler. Sandsynlighedsfunktionen er afhængig på følgende måde $P(x_i|x_{i-1})$ hvor x_{i-1} er de konkrete forgående punkter i den forgående iteration [6].

Nedenstående trin er derfor involverede i at lokalisere robotten på kortet.

1. n tilfældige konkrete stokastisk variabler x_i bestemmes for kun første iteration.
2. Robotten laver en måling m på formen (r, g, b) på dens nuværende position.
3. Målingen m sammenlignes nu med x_i på følgende måde $|m - x_i|$
4. Et nyt sandsynlighedsfelt λ der er magen til det forgående hvor sandsynlighedsfunktionen er afhængig af fejlen, således at $P(x_i||m_{-1} - x_{i-1}|)$.
5. Der bestemmes nu λ_n tilfældige punkter fra λ, hvor de punkter med højste fejl har den højeste sandsynlighed, hvor $\lambda_n < n$, og fjernes fra λ_U
6. Robotten bevæger sig nu til en ny flise i en hvilken som helst retning, hvor robotten stadig vil være inden for fliserne. Alle resterende punkter i λ flyttes i samme retning.
7. Der bestemmes nu λ_n, altså samme antal som blev fjernet, punkterne fra et sandsynlighedsfelt hvor sandsynlighed er størst for de med den laveste fejl, altså så punkter med en lav fejl har en højere sandsynlighed. En af de omliggende fliser fra dette punkt som robotten kunne have bevæget sig over vælges og tilføjes til udfaldsrummet λ_u. Derefter gentages de forgående tre trin det antal gange robotten bevæger sig.

Hvis overstående trin følges, vil man starte ud med en masse partikler spredt jævnt hen over udfaldsrummet U, som betegner kortet. Men som processen gentages bliver der dannet en koncentration af partikler der hvor robotten faktisk befinder sig. Nedenstående figurer illustrerer dette, hvor de sorte punkter er partikler og det orange punkt er robotten.



I det følgende diskussions afsnit vil opgaven diskutere muligheder og begrænsninger i forbindelse med anvendelsen af Monte Carlo Lokalisering i en konkrete autonom robot.

5 Diskussion

En begrænsning af Monte Carlo Metoden er, at den ikke kan bruges til at lokalisere en robot i omgivelser der er dynamiske. Dette skal forstås på den måde, at hvis robottens interne kort ikke længere stemmer overens med hvad sensorne måler vil robotten ikke kunne tilnærme sig sin lokalisering. Dette vil altså betyde at, hvis robotten måler afstanden til vægge vil den for eksempel blive forvirret når mennesker bevæger sig omkring den. Den vil her forveksle menneskerne med statiske murer, og tro at den er meget tættere på en mur end den i virkligheden er [6].

En anden begrænsning ved Monte Carlo Lokalisering er Robot-kidnapnings problemet. Monte Carlo Lokalisering i forhold til dennes implementering kan både være robust og ikke robust over for dette problem. Robot kidnapnings problemet består i at robotten for eksempelvis bliver samlet op og flyttet til et nyt punkt uden at blive informeret omkring dette [6]. Jeg vil mene, at dette problem kan løses ved at introducere interferens i form af et antal partikler, som er symmetrisk fordelt hen over udfaldsrummet. Dette vil sandsynligvis have den effekt, at hvis robotten blev flyttet til et nyt punkt, ville interferens give mulighed for at de kommende iterationer kunne danne en koncentration omkring disse partikler, som ikke er i nærheden af den før hen tilnærmede lokalisering. Altså vil interferens fungere som om en form for kickstart.

Et tredje eksempel på en begrænsning eller en mulighed i forbindelse med Monte Carlo lokalisering er robotter der bevæger sig i rummet. Hvis vi betragter en robot, der skal lokalisere sig selv i rummet i stedet for i planet skal der bruges et større antal partikler. Dette vil betyde, at hvis man ønsker at nedbringe fejlen af aproximering med en faktor 10 skal man øge antallet af partikler med 10^3 i modsætning til planet, hvor man skal bruge 10^2 partikler. At simulere denne mængde partikler kan være en begrænsning for de små microprocessor, der driver vores robotter, men i takt med at vi udvikler mindre og hurtigere processorer vil det blive muligt at øge antallet af partikler der simuleres samtidigt [6].

Desuden er Monte Carlo Lokaliserings præcision afhængig af, hvor præcist vi kan bestemme fejlen mellem vores interne kort, og derved partiklerne, og de målinger vores sensorer registrerer. Fejl her kunne eksempelvis skyldes, at en afstandssensor har en nøjagtighed på $\pm 1cm$. Med et stort nok antal partikler ville dette problem forsvinde da vi udvælger en håndfuld partikler, der har den lavest fejl, og derved overlever kun de partikler med de mindste fejl. For tilfældet med en farve sensor kan fejlen skyldes at målingerne kan variere med eksempelvis op til 10%.

6 Konklusion

Monte Carlo Metoden blev udviklet under anden verdens krig, og siden da er metoden blevet adopteret til at kunne løse forskellige problemer, der er komplekse eller tilfældige i deres natur.

I denne opgave er der blevet redegjort for, hvordan Monte Carlo Metoden kan anvendes i praksis og teoretisk samt hvorledes en autonom robot kan implementere en sådan algoritme til at lokalisere sig selv i kendte omgivelser. Mulighederne for Monte Carlo Metoden er mange, men på grund af at antallet af simuleringer, der skal til for at reducere fejlen, da det ikke er proportional med reduktionen af fejlen. Dette betyder, at for at reducere fejlen med en faktor 10 skal antallet af simuleringer i nogle tilfælde øges med en faktor 100.

I afsnittet om stokastiske variabler blev der fundet frem til en række egenskaber, her i blandt den forventede værdi. Dette kunne bruges til at beskrive Monte Carlo Metoden og hvorledes en række eksperimenter og simuleringer kan udføres.

Monte Carlo Metoden kan benyttes inden for mange områder, heriblandt kan metoden Hit-or-Miss bruges til at integrere en figur eller funktion, der ikke kan udledes algebraisk. I opgaven blev metoden anvendt til at integrere enhedscirklen og derved udlede π.

I afsnittet om anvendelsen samt diskussionen blev Monte Carlo Lokalisering beskrevet samt .

Monte Carlo Metoden kan også anvendes til at lokalisere en autonom robot, og kan gøre dette forholdsvist robust.

Bibliografi

[1] Kasper K. Berthelsen. „Note om Monte Carlo metoden“. I: (2014).

[2] Povl Lebeck Ølgaard. *Manhattanprojektet*. 2017. URL: http://denstoredanske.dk/index.php?sideId=121576.

[3] Atomic Heritage Foundation. *Computing and the Manhattan Project*. 2014. URL: https://www.atomicheritage.org/history/computing-and-manhattan-project.

[4] Daniel Kjær. „Sandsynlighedsbaserede metoder [Monte Carlo-metoden]“. I: (2013).

[5] G.W. Lucas. *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators*. 2000. URL: http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html.

[6] Sebastian Thrun. „Particle Filters in Robotics“. I: (2002).

[7] toptal. *Dubble Sort*.

[8] Nitschky Schmidt Vestergaard. *Statistik C*. Systime, 2008.