Ruteplanlægning til autonome plæneklippere

P2-projekt 2006:
Ruteplanlægning til autonome plæneklippere

Vejleder:
Erling V. Matthiesen

Bivejleder:
Jette E. Holgaard

Gruppe C226:
Søren Hede
Christoffer H. Poulsen
Claus Pedersen
Jesper E. Pedersen
Jesper Skjødt
Søren Thorup

Aalborg Universitet
Det Teknisk-Naturvidenskablige Basisår
Elektronik & Datateknik

Aalborg Universitet - Strandvejen 12-14 - 9000 Aalborg - Tlf. 96 35 80 80 - Telefax 98 13 63 93

Det Teknisk-Naturvidenskablige Basisår

Projektgruppe: A302

Titel: Ruteplanlæging til autonome plæneklippere
Tema: Modellernes virkelighed
Projektperiode: P2, 1. februar - 29. maj

Projektgruppe: A302
Linje: Elektronik og Datateknik

Deltagere: Søren T. Hede
Christoffer H. Poulsen
Claus T. B. Pedersen
Jesper E. Pedersen
Jesper Skjødt
Søren Thorup

Vejleder: Erling V. Mathiesen
Bivejleder: Jette E. Holgaard

Oplagstal: 14
Sidetal: 89
Afleveret: 29/05/2006

Synopsis:

Formålet med denne rapport er, at undersøge ruteplanlægning for autonome plæneklippere, og hvilke muligheder for forbedringer, der kan være.
I problemanalysen har vi undersøgt de nuværende autonome plæneklippere, og har fundet ud af at nuværende autonome plæneklippere ikke benytter et effektivt ruteplanlægningssystem, men blot kører tilfældigt rundt på græsplænen. Vi har endvidere lavet en behovsanalyse, og vist at der er basis for et forbedret ruteplanlægningssystem, og vi har undersøgt, hvordan forskellige rutealgoritmer fungerer.
Vi har derfor, med SPU modellen, udviklet et program, til at simulere en autonom plæneklippers kørsel med forskellige rutealgoritmer, og til at beregne rutealgoritmernes effektivitet på et givet areal. Derefter har vi brugt programmet til at teste forskellige rutealgoritmer på forskellige arealer. Formålet med disse test har været at undersøge, om nogle af algoritmerne er mere effektive end andre, med henblik på at kunne vise, at de nuværende produkters kørselssystem vil kunne forbedres, og hvor meget de kan forbedres.

Forord

Denne rapport er udarbejdet i forbindelse med et P2 projekt i perioden 1. februar til 29. maj ved den datatekniske linie på Aalborg Universitet under navnet ”Ruteplanlægning til autonome plæneklippere”. Temaet for semestret er ”Modellernes Virkelighed”.

Oprindelig troede vi, at vores løsning ville være baseret på kørsel med en LEGO MindStorms model, men grundet begrænset tid, har vi valgt udelukkende, at simulere kørsel med rutealgoritmer

i et program.

I forbindelse med dette projekt vil vi gerne takke Teknisk Forvaltning ved Aalborg Universitet, for deltagelse i interview.

Bagest i rapporten er vedlagt appendiks med relevante informationer om dele af rapporten som fx interview guide og matlabscript samt en cd med internetkilder og vores simuleringsprogram.

Læsevejledning

Rapporten er bygget op på følgende måde:

- **Indledning:** Kap. 1
- **Problemanalyse:** Kap. 2
- **Problemløsning:** Kap. 3 - kap. 8

Igennem hele rapporten er kildehenvisninger angivet med [], hvor fx [4] refererer til den fjerde kilde på litteraturlisten i appendikset. Bestemte termer og udtryk er forklaret nærmere i fodnoter, hvor vi har fundet det nødvendigt. I appendikset er der desuden en definitionsliste over de pågældende udtryk.

For at få mest muligt ud af denne rapport er det et krav, at have kendskab til programmeringssproget C.

Søren T. Hede

Christoffer H. Poulsen

Claus T. B. Pedersen

Jesper E. Pedersen

Jesper Skjødt

Søren Thorup

Indhold

1 Indledning **7**

1.1 Initierende problem 7

2 Problemanalyse **9**

2.1 Indledning 9

2.2 Produkter på markedet 9

2.3 Behovsanalyse 13

2.4 Rutealgoritmer 15

2.5 Matlab test af Random Direction 21

2.6 Positionering 22

2.7 Produktvision 28

2.8 Konklusion 28

2.9 Problemformulering 28

3 SPU modellen **29**

3.1 SPU udviklingsmodellen 29

4 Kravspecifikation **31**

4.1 Indledning 31

4.2 Generel beskrivelse 31

4.3 De specifikke krav 34

4.4 Eksterne grænsefladekrav 36

4.5 Krav til programmets ydelse 36

4.6 Kvalitetsfaktorer 37

5 Design **39**

5.1 Programdesign 39

5.2 Procesdesign 39

5.3 Moduldesign 41

6 Test **57**

6.1 Modultest 57

6.2 Modulintegration 59

6.3 Procesintegration 60

6.4 Accepttest 60

7 Implementation **61**

7.1 Programbeskrivelse 61

7.2 Testkørsel 62

7.3 Forbedringsmuligheder 64

8 Forsøg **65**

9 Konklusion **73**

9.1 Perspektivering 74

A Definitioner **77**

B Ordliste **79**

C Interviewguide **81**

D Matlab-script **83**

E Kildekritik **87**

Litteratur **90**

F Bilag **91**

F.1 CD 91

Kapitel 1

Indledning

Dette projekt omhandler en af de forbedringsmuligheder, der er på de nuværende autonome plæneklippere, som eksisterer på det danske marked anno 2006. Det område som vores projekt omhandler, er plæneklippernes ruteplanlægning. Som baggrund for dette, antager vi, at de nuværende autonome plæneklippere kører over plænen i en tilfældig rute, og at de ikke ved, hvor de har været. Vi mener, dette resulterer i ressourcespild og forlænger den tid, det tager at slå plænen. Begrundelsen for dette er, at plæneklipperen kommer til, at køre over det samme sted på plænen flere gange, for at sikre sig, at de har dækket hele græsarealet.

Vi vil endvidere komme ind på effektiviteten af to af de nuværende autonome plæneklippere, og udviklingen af et systematisk navigationssystem til autonome plæneklippere. Med effektivitet menes her, at vi vil finde den kortest mulige rute for plæneklipperen. Fremgangsmåden er, at bearbejde de informationer, som plæneklipperen indsamler under opstart. og derved systematisk beregner en rute til plæneklipperen, så den slår plænen, uden at køre over det samme sted mere end en gang.

1.1 Initierende problem

Hvordan kan autonome plæneklipperers ruteplanlægning gøres mere effektiv?

1. Hvilket kørselssystem benytter de eksisterende produkter?
2. Hvor effektive er de eksisterende produkter mht. dækning af et areal?
3. Hvilke problemer giver de eksisterende produkters bevægelsesmønstre?
4. Ser de eksisterende brugere af autonome plæneklippere de usystematiske plæneklippere som et problem?
5. Hvilke rutealgoritmer kan bruges til, at slå græsplænen i et system, hvor plæneklipperen i så vidt muligt omfang ikke kører over det samme sted mere end højest nødvendigt?

For at undersøge disse spørgsmål har vi valgt følgende fremgangsmåde:

Først undersøger vi de eksisterende produkter på markedet. Dette gøres for at undersøge, om vores antagelse, om at autonome plæneklippere ikke er systematiske, er korrekt, samt undersøge spørgsmål 1 og 2. (Afsnit 2.2)

Herefter undersøger vi spørgsmål 4, ved at udspørge eksisterende brugere af autonome plæneklippere. For at undersøge om de oplever problemer med, at de ikke er systematiske og derefter undersøge, om de er interesserede i vores idé forbedring. (Afsnit 2.3)

Til sidst undersøger vi ud fra spørgsmål 5, om det er muligt, at forbedre systemet, ved at bruge en rutealgoritme, som er mere systematisk. Vi vil også undersøge, om der er nogle specielle krav, der skal opfyldes, for at kunne anvende algoritmen. Dette kan f.eks. være at plæneklipperens position og området skal være kendt. (Afsnit 2.4)

Kapitel 2

Problemanalyse

2.1 Indledning

Problemanalysen er udarbejdet med henblik på, at afdække behovet for mere effektive typer autonome plæneklippere og undersøge, hvordan det kan lade sig gøre, at forbedre autonome plæneklippere. Vi har valgt at opdele problemanalysen i fem underafsnit, som hvert afsnit afsluttes ved at perspektivere og konkludere resultatet.

- **Produkter på markedet**
  Under "Produkter på markedet" undersøger vi de nuværende produkter på markedet. Undersøgelsen er hovedsageligt en perspektivering i de nuværende produkters energiforbrug samt rutealgoritmer, priser, klippebredde, og arealeffektivitet. Kapitel 2.2
- **Behovsanalyse**
  I afsnittet "Behovsanalyse" undersøger vi, om der er et behov for vores ide, ved at interviewe Teknisk Forvaltning, Aalborg Universitet. Kapitel 2.3
- **Rutealgoritmer**
  I afsnittet "rutealgoritmer" undersøger vi nogle forskellige rutealgoritmer mht. forbedringsmuligheder til vores produktvision. Kapitel 2.4
- **Positionering** I dette afsnit vil vi undersøge hvilke metoder, der kan benyttes til at bestemme positionen på en autonom plæneklipper. Dette kan være en nødvendighed i forhold til en rutealgoritme. Kapitel 2.6
- **Produktvision**
  I "Produktvision" beskriver vi, på baggrund af vores overordnede vision, omhandlende en forbedring af kørselsmønstret. Kapitel 2.2 - 2.6

2.2 Produkter på markedet

I dette afsnit undersøger vi de forskellige produkter på markedet, for at få et overblik over, hvad de kan i forvejen.

De to største producenter af autonome plæneklippere på det danske marked i dag (april 2006) er Husqvarna/Electrolux og FriendlyRobotics. Vi sammenligner kun de to producenter imellem, da vi antager, at de er de eneste, der har en nævneværdig markedsandel på det danske marked. Det er ikke lykkedes os at finde andre produkter, og vi har forsøgt at dokumentere dette ved søgning på Google[1]. Ydermere har vi forsøgt, at underbygge vores antagelse med statistikker, men det har ikke været muligt at fremskaffe. Til sidst har vi undersøgt hvilke produkter, der bliver solgt i de

[1] Vi har søgt på følgende ord: intelligent græsslåmaskine, intelligent plæneklipper, førerløs plæneklipper, førerløs græsslåmaskine, robot plæneklipper, robot græsslåmaskine, autonom plæneklipper og autonom græsslåmaskine

store byggemarkeder som Silvan, Jem & Fix og Bauhaus, men de sælger desværre ikke denne type maskineri.

Vi har valgt, at sammenligne producenternes topmodeller, da det er disse, som har flest funktioner og features. Det er Husqvarnas Auto Mower og FriendlyRobotics Robomow RL1000. Vi vil sammenligne de to produkter på følgende parametre:

- Positioneringsudstyr
- Batteritid
- Strømforbrug
- Pris

Vi har taget kontakt til importørerne, for at få mere detaljeret viden om positioneringsudstyret på de to modeller. De anbefalede os, at skrive en email til producenterne, for at få flere detaljer omkring hvordan positioneringen virker. Dog har ingen af de to firmaer svaret tilbage på vores forespørgsler, og vores viden omkring plæneklipperne er derfor baseret på manualerne til disse, som dog kun kan bidrage med begrænsede informationer.

2.2.1 Positioneringsudstyr

- **Robomow**
  Robomow anvender et navigationssystem baseret på et elektronisk kompas, der indstiller sig efter jordens magnetiske poler. Robomow skal kalibreres første gang den bruges. Det har ikke været muligt, at få præciseret den præcise funktion af kompasset fra producentens side.

- **Auto Mower**
  Auto Mower kører efter en algoritme, hvor Auto Mower kører indtil den støder på noget, herefter vælger Auto Mower en ny retning indenfor en 180 graders spændvidde, og følger herefter denne, indtil den støder på et nyt punkt. Hvis man vil have Auto Mower til at bevæge sig i en snæver passage, så medfølger der et guide kabel, som Auto Mower kan følge til/fra ladestationen.

2.2.2 Batteritid og arealdækning

Der er flere faktorer, der spiller ind, når batteritiden skal fastsættes. En væsentlig faktor er havens udformning, antal bede og snævre passager. Andre vigtige faktorer kan være knivenes tilstand og plænens fugtighed. Tiderne i disse sammenligninger er blevet oplyst fra producenterne[2][3], og kan derfor afvige, hvis ikke der har været opstillet de samme forudsætninger for beregningerne. I sammenligningen vælger vi konsekvent den maksimale værdi, der er oplyst fra begge produkter. at sammenligne på, da vi ikke har mulighed for selv at teste de oplyste tal. Ifølge FriendlyRobotics klipper R1000 i gennemsnit 150 m^2 i timen. Den klipper omkring 1,5 - 3 timer ad gangen og skal derefter lade i mindst 16 timer.
Auto Mower kan i gennemsnit klippe 75 m^2 i timen. Auto Mower klipper omkring 40 - 60 minutter ad gangen og skal derefter lade i 45 minutter.
Hvis vi sammenligner klippet areal pr. cyklus (en cyklus defineres som tiden for én opladning samt én klipning), slår Auto Mower maks. 75 m^2 pr. cyklus, hvorimod Robomow kan klare 450 m^2 pr. cyklus.
Da de to produkters cykler ikke varer lige lang tid, opstiller vi en ligning, for at kunne sammenligne klippet areal pr. døgn.

$$\frac{1 \text{ døgn}}{1 \text{ cyklus}} \cdot \text{areal pr. cyklus} \;=\; \text{areal pr. døgn} \tag{2.1}$$

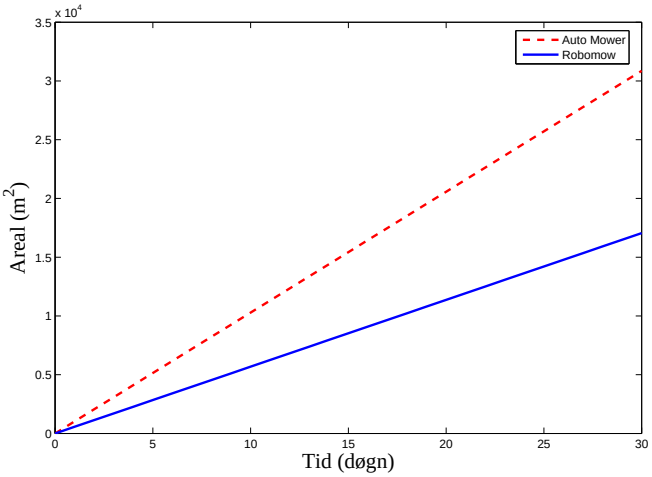
Kigger vi på pr. døgn, kan Auto Mower klare 1028,57 m^2.

$$\frac{24 \text{ timer}}{1,75 \text{ timer}} \cdot 75m^2 = 1028,57m^2 \tag{2.2}$$

Hvor Robomow kan klare 900 m^2, da den maksimalt kan komme til at klippe 2 gange om dagen, men hvis vi udregner gennemsnittet på samme måde som før, får vi cirka en forskel på en faktor 2

$$\frac{24 \text{ timer}}{19 \text{ timer}} \cdot 450m^2 = 568,42m^2 \tag{2.3}$$

Som det fremgår af førnævnte beregninger, kan Auto Mower slå næsten dobbelt så stort et areal pr. døgn som Robomow.



Figur 2.1 *Sammenligning af de to plæneklipperes dækkede areal i forhold til tid*

2.2.3 Pris

Der er ikke den store prisforskel på de to modeller. Forskellen ligger på 50 kroner, så det er ikke på det sammenligningsgrundlag, at man skal vælge sin model. Prisen har kun noget at sige, når man er på det stadie, hvor man stadig overvejer, om man skal have en autonom plæneklipper. Prisen på de 15.000 kroner kan virke som meget for en privatperson, og derfor er den mest egnet til virksomheder, hvor man f.eks. vil have slået græs på større områder, der er svært tilgængelige for større maskineri. Hvis man vil se, om man kan tjene investeringen ind igen ved at spare mandetimer, så skal man samtidigt have for øje, at plæneklipperen er dyr i driftomkostninger pga. flere forskellige parametre som f.eks. konstruktionens opbygning[2]. De forskellige faktorer gør derfor regnestykket meget kompliceret og tidskrævende, og derfor vil vi ikke gå nærmere ind på det.

2.2.4 Delkonklusion

Vi har i vores initierende problem stillet følgende spørgsmål:

1. Hvilket kørselssystem benytter de eksisterende produkter?
2. Hvor effektive er nogle af de eksisterende produkter mht. dækning af et areal?

Af de to modeller af autonome plæneklippere, vi har kigget på, har ingen af de to noget effektivt ruteplanlægningssystem, men benytter sig af tilfældig kørsel, hvor de skifter retning, hver gang de kommer til kanten af græsplænen, og derved vil de få dækket hele arealet over en given periode. Desuden har vi beregnet, hvor stort et areal de to modeller kan klippe pr. døgn og sammenlignet de to resultater. Man vil også kunne forbedre plæneklippernes effektivitet, ved at gøre klippebredden og/eller batteriet større. Dette vil give ekstra omkostninger i form af delkomponenter, og plæneklipperen vil alligevel kunne effektiviseres ved hjælp af et bedre ruteplanlægningssystem.

Til sidst har vi lavet en sammenligningstabel, der giver et hurtigt overblik over sammenligningerne.

Sammenligning	**Robomow**	**Auto Mower**
Pris	14.949,-	14.995,-
Positionering	Elektronisk kompas	Vilkårlig vinkel
Batteritid	1 - 3 timer	40 - 60 minutter
Klippebredde	56 cm.	22 cm.
Strømforbrug	Maks. 20 kWt/måned ved et arbejdsområdet på 1800 m^2.	20 kWt/måned ved maks. klippeområde[3]

[2] Se 2.3.5

[3] Producentens egen formulering. Det maksimale klippeområdes størrelse er ikke oplyst

2.3 Behovsanalyse

Vi ønsker at lave en behovsanalyse for at finde ud af, om der overhovedet er et behov for vores løsning. Behovsanalysen er baseret på markedsanalyse-modellen og vi ønsker, at anvende den til at lave et kvalitativt interview. Processen er inddelt i 5 faser:

1. **Problemformulering**
   I denne fase præciseres hvilket problem eller mulighed, som skal undersøges i markedsanalysen.
2. **Analysemålsætning**
   Beskriver hvad vi vil have svar på med vores analyse.
3. **Værdien af informationen**
   Hvor mange ressourcer der bliver brugt på at foretage behovsanalysen? For eksempel opgjort i tid og penge. Vi vil dog ikke benytte dette punkt i vores analyse.
4. **Analysedesign**
   I denne fase bestemmes det, hvordan analysen skal gennemføres. Bl.a. analysetype, informationskilder, og design interviewguide.
5. **Implementering**
   Her skal dataene indsamles og analyseres, så man kan drage en konklusion på det problem og de spørgsmål, der er blevet fremlagt i problemformuleringen og analysemålsætningen.

2.3.1 Problemformulering

De typer autonome plæneklippere som i dag findes på markedet, anvender en teknik som gør, at de kører i tilfældige mønstre. Dette resulterer i ineffektivitet, og vi ønsker derfor, at undersøge behovet for en forbedring af effektiviteten. Endvidere ønsker vi at finde ud af hvordan nuværende løsninger fungerer i praksis.

2.3.2 Analysemålsætning

Vi ønsker, at få belyst problemstillinger i forbindelse med autonome plæneklippere. Herunder ønsker vi at finde ud af om der med nuværende løsninger, er problemer med effektiviteten og om der er en interesse for forbedring. Dertil vil vi gerne have svar på følgende analysespørgsmål:

- Er der behov for forbedring af plæneklippernes navigationssystem?
- Giver de nuværende plæneklipperes navigationssystem problemer?

I forbindelse med fremstillingen af spørgsmålene til analysen opstiller vi følgende hypoteser:

- Private forbrugere er ikke interesserede i en forbedring
- En forbedring må ikke gå udover betjeningen af produktet

For at få konstruktive svar, vil vi henvende os til nogen, der i forvejen benytter autonome plæneklippere. Vores produkt omhandler forbedring af effektiviteten, og derfor vil det være mest relevant, at afgrænse sig til eksisterende brugere. Vi vælger at tage kontakt til Teknisk forvaltning ved AAU, da de dagligt arbejder med autonome plæneklippere.

2.3.3 Værdien af informationen

Da Teknisk forvaltning på AAU har autonome plæneklippere, og dagligt arbejder med dem, har vi fået en masse værdifuld førstehåndsviden. Interviewet er derfor vores primære informationskilde.

2.3.4 Analysedesign

Da vi vil undersøge et potentielt problem i form af manglende effektivitet for autonome plæneklippere ,og vi ønsker at få uddybende informationer om andre beslægtede problemer, vil vores analyse være eksplorativ.
For at få et så pålideligt analyseresultat som muligt, er vi nødt til at benytte primære informationskilder. En anden årsag til dette er, at vi ikke har mulighed for, at finde eksisterende analyser der belyser netop vores problemstilling. Desuden vil vi lave analysen som et interview, da vi derved har bedst mulighed for at få uddybet problemstillingerne.
Vi har valgt at lave interviewet med gartner Jørn Gregersen fra Teknisk Forvaltning på Aalborg Universitet. Udfra vores analysemålsætning og den baggrundsviden vi har om emnet, har vi udformet en interviewguide. Spørgsmålene i interviewguiden er i Appendix C.

2.3.5 Implementering

Gennem vores vejledere kom det os til kende at Teknisk Forvaltning, AAU, har autonome plæneklippere i brug. Derfor var det interessant for os at interviewe Teknisk Forvaltning i forbindelse med analysen.

Vi var derfor interesseret i, at lave et interview med teknisk forvaltning, for at få førstehåndsviden omkring disse plæneklippere. Blandt andet ville vi finde ud af, hvordan plæneklipperne fungerer i praksis, hvordan en løsning til vores problemformulering kan bruges, og på hvilken måde det vil kunne forbedre plæneklipperne. Desuden ville vi også kunne bruge informationerne om plæneklipperen til vores produktsammenligning og markedsanalyse, og til en kravspecifikation til vores løsningsprogrammel.

Vi ringede og aftalte et møde med overgartneren på AAU. Før interviewet havde vi lavet en interviewguideC, som vi ville have svar på. Vi holdte os ikke kun til de spørgsmål, vi havde lavet inden interviewet, da en del af de svar vi fik, rejste nye spørgsmål. Dette var en fordel ved den valgte metode, da vi kunne få svar på spørgsmål, der ellers kunne være dukket op senere i projektforløbet, og fordi vi nemt kunne have overset nogle spørgsmål, som vi senere ville have svar på.

2.3.5.1 Informationer tilegnet under interviewet

Teknisk Forvaltning, AAU, har 6 Husqvarna Auto Mower plæneklippere, og dobbelt så mange installationer med ladestation. Teknisk Forvaltning havde først en enkelt plæneklipper på prøve i 1999, som de havde gode erfaringer med, hvorefter de anskaffede sig flere. De bruges på universitetet til at klippe græsset i de lukkede gårdarealer. Hver plæneklipper dækker et areal på ca 600 til 700 $m2$ og flyttes mellem gårdhaverne ca hver 2. og 3. dag eller efter behov (der er flere gårde end plæneklippere).

Årsagen til anskaffelsen er, at de før var nødt til at bære store konventionelle græsslåmaskiner gennem bygningerne, for at kunne komme til at slå græsset i gårdhaverne. Det var meget besværligt og krævede ekstra rengøringsarbejde og larmede. Auto Mower løser dette problem i kraft af, at dens elmotor er forholdsvis lydsvag, og at den kun vejer ca 8 kg og kan medbringes næsten overalt. Ud over dette klipper den også græsset pænere, da den gennemkører arealet mere end en gang. Personalet hos teknisk forvaltning skal endvidere kun bruge den halve tid på at få græsset slået, og de slipper for besværet med at bære en konventionel græsslåmaskine gennem universitetets bygninger.

Af oplysninger som ikke er oplyst fra producentens side er, at Auto Mower ikke kan slå helt ud til kanten af græsset ved for eksempel mure, hvilket kræver at der klippes efter af en person. I praksis

skal der også føres tilsyn med Auto Mower'en mindst en gang om dagen, da der nemt kan opstå uforudsete komplikationer, så som at den ikke kan finde tilbage til ladestationen, eller at den er blevet afbrudt i sit arbejde af en nysgerrig forbipasserende el.lign.

I følge teknisk forvaltning er Auto Mower's største mangler, at:

- Den ikke har display til aflæsning af fejlkoder.
- Det er en spinkel konstruktion.
- Den er dyr i indkøb og vedligeholdelse.
- Dens hjul står ikke godt fast på vådt græs.

2.3.6 Interview konklusion

Igennem interviewet har vi belyst fordele og ulemper ved autonome plæneklippere. Det kan vi bruge til, at besvare vores 3. spørgsmål i vores initierende problem

- Hvilke problemer giver de eksisterende produkters bevægelsesmønster?

Ifølge teknisk forvaltning, er der ikke direkte behov for en løsning, da deres arbejdsgang er tilpasset det, som deres autonome plæneklippere på nuværende tidspunkt kan. Derfor har de heller ikke problemer med deres autonome plæneklipperes bevægelsesmønster. Dog vil en løsning kunne formindske den tid, som Auto Mower bruger på dens arbejde, hvilket i AAU's tilfælde betyder, at den kan flyttes til næste gård hurtigere end 2-3 dage, hvilket vil kunne spare dem for nogle plæneklippere, og dermed også driftomkostninger.

2.4 Rutealgoritmer

I dette afsnit beskrives forskellige rutealgoritmer, som vi har valgt at dele op i to kategorier: Tilfældighedsbaserede algoritmer og planlægningsalgoritmer. Tilfældighedsalgoritmer er karakteriseret ved, at der indgår parametre med tilfældigt variende værdier i rutegenereringen, der derudover foregår løbende. Planlægningsalgoritmerne er modsat tilfældighedsalgoritmerne karakteriseret ved, at ruten planlægges på forhånd eller på forhånd kan forudsiges. Givet vores problemstilling har de i dette afsnit nævnte algoritmer til formål, at dække et givet areal.

Tilfældighedsalgoritmer bruges typisk i områder med ukendte grænser, mens begge typer tilfældigheds- og planlægningsalgoritmer kan bruges i kendte områder. Det ville også være problematisk at bruge planlægningsalgoritmer i et ukendt område, da der kan være uforudsete forhindringer. Begge kategorier kan bruges til forskellige formål. Nogle rutealgoritmer fra begge kategorier, kan dog bruges til samme formål. Arealdækning kan f.eks. både gøres ved at bevæge sig tilfældigt rundt på et areal, og ved at følge en forud planlagt rute. Nogle tilfældighedsbaserede algoritmer er dog langt simplere at implementere, fordi de ikke kræver positionering. De vil derfor ofte være en billigere løsning. Det vides dog ikke med sikkerhed, hvornår hele området er dækket. En planlægningsalgoritme derimod giver en fast rute, der kan reproduceres og er tilregnelig mht. distance og tid.

Ideen med at analysere algoritmerne er, at undersøge hvordan de forskellige algoritmer virker. Måden, vi beskriver dem på, er at gennemgå, hvordan hver algoritme virker teoretisk, og hvad fordelene og ulemperne er med dem mht. vores mål om at dække et areal på en effektiv måde, forstået sådan at arealet dækkes hurtigst muligt. Nogle algoritmer bygger på tilfældighedsparametre, mens andre genererer en rute, der så vidt muligt sørger for, at den ikke krydser sig selv, altså at ruten ikke løber over det samme punkt i området.

Nogle algoritmer er lavet til, at løse Traveling Salesman-problemet, som går ud på, at man skal passere en række fastsatte punkter i den billigst mulige rækkefølge. Billigst vil i nogle tilfælde betyde kortest. Som problemets titel indikerer, kan problemet beskrives med en allegori, hvor en handelsmand skal besøge en række byer præcis én gang, ved at følge den billigst mulige rute,

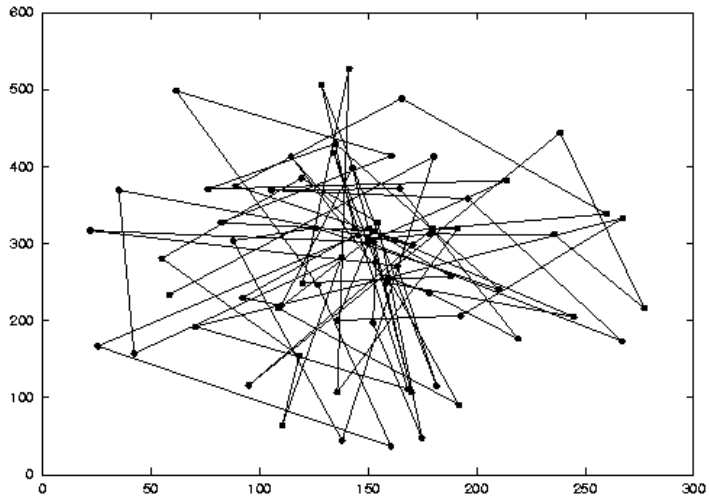
hvor afstanden mellem hver by har en bestemt værdi. Placeres punkterne på en strategisk måde i et område, kan disse algoritmer også bruges til, at finde den korteste vej, der dækker området.

Følgende algoritmer er beskrevet ud fra Mobility models [4] og analyseret i dette kapitel:

1. Tilfældighedsbaserede algoritmer:
    - Random Walk
    - Random Direction
    - Random degree af Gauss-Markov
    - Random Waypoint
2. Planlægningsalgoritmer
    - Genetic Algorithm
    - Distance Transform Path Planning

2.4.1 Random Walk

Algoritmen Random Walk går i alt sin enkelthed ud på, at der vælges en tilfældig retning og hastighed i en fast defineret tid eller distance. Både hastigheden og retningen er forudbestemt ud fra en tabel med minimum hastighed, maksimum hastighed og en vinkel mellem 0 og 360 grader. Hvis enheden møder områdets ydre grænse i det givne interval, vil den automatisk dreje ind i området med samme udfaldsvinkel som indfaldsvinkel og fortsætte med samme hastighed, indtil tidsintervallet udløber eller distancen er nået.

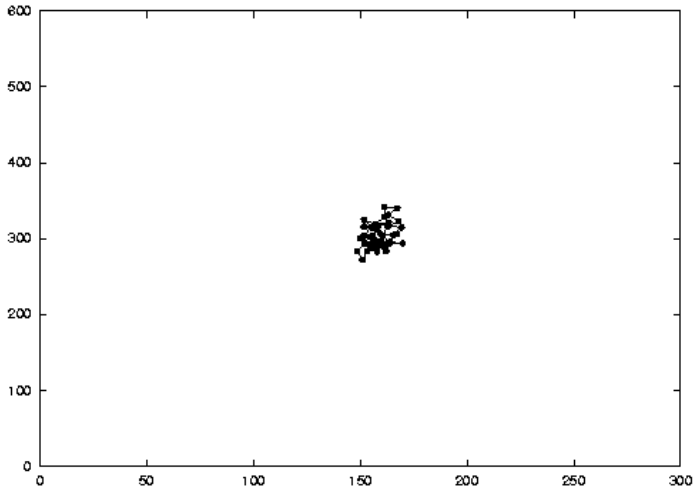


Figur 2.2 *Random walk-modellen[4]. Her er tidsintervallet, hvori der rejses, før der skiftes retning og hastighed, konstant. Bemærk at afstanden mellem punkterne ikke er lige store, fordi hastigheden skifter fra strækning til strækning*

Fordele

- Kan bruges i områder med og uden ydergrænse.
- Kræver ingen viden om området udover ydergrænserne, hvis det bruges.
- Vil med tiden dække hele området.

Ulempe



Figur 2.3 *Random walk-modellen[4]. Her er afstanden mellem punkterne konstant. Dvs. at tidsintervallet, hvori der rejses, er afhængigt af hastigheden.*

- Den bevæger sig over det samme område flere gange.
- Tiden det tager, at dække arealet er uforudsigelig.
- Den bruger meget tid på at stoppe og dreje.
- Har svært ved at komme ud i hjørnerne af et givent område.

2.4.2 Random Direction

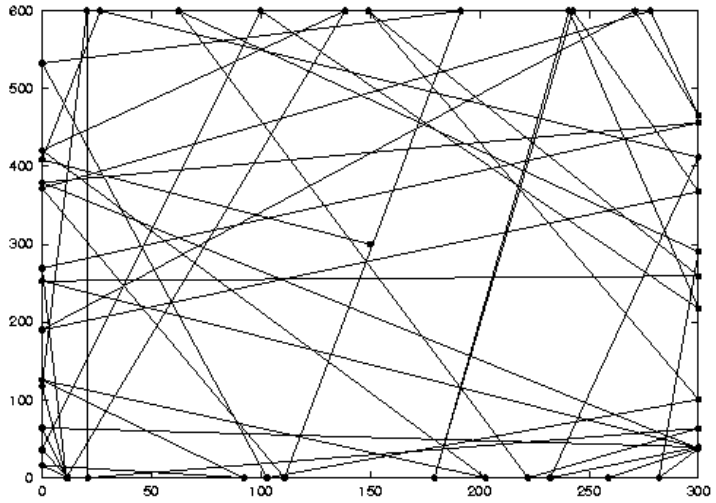
Random Direction algoritmen er defineret, ved automatisk at skifte til en ny retning indenfor et afgrænset området med en vinkel mellem 0 og 180 grader, når kanten af området nås. Her holdes derudover en foruddefineret pause. Denne algoritme er et forsøg på, at få en større udspredning af punkterne, hvor der skiftes retning, og på strækningerne. Dette skal således formindske, at det samme område overkøres flere gange, mens andre sjældent bliver overkørt. Der er udviklet flere forskellige afarter af algoritmen. Eksempelvis er ”Modified Random Direction Mobility Model” en sammenblanding af Random Direction og Random Walk, hvor man udover at vælge en ny retning ved hver ydre grænse, også bestemmer en afstand, man ønsker at bevæge sig. Det resulterer i, at man automatisk vil stoppe op og begynde at bevæge sig ad en ny retning, selvom afstanden er kortere end afstanden til ydergrænsen.

Fordele

- Kører fra kant til kant og derved bliver strækningerne så lange som muligt.
- Er bedre til at komme helt ud til kanten end Random Walk.
- Vil før eller siden bevæge sig over hele området.
- Kræver ingen viden om området udover ydergrænserne.

Ulemper

- Tiden, det tager at dække arealet, er uforudsigelig.
- Den vil køre over de samme steder mange gange, før den har dækket hele området, hvilket giver ressourcespild,.



Figur 2.4 *Random direction-modellen*

2.4.3 Random degree af Gauss-Markov

I modsætning til tidligere nævnte tilfældige algoritmer tager Gauss-Markov modellen højde for tidligere værdier af hastighed og retning. Da disse får indflydelse på de nye værdier, undgår man pludselige stop og skarpe sving, som man typisk vil få med Random Walk og Random Direction modellerne. Gauss-Markov er dog oprindeligt lavet med henblik på at simulere netværk og ikke at dække et areal. Gauss-Markov modellen bruger én parameter til at variere graden af tilfældighed i mønstret. Med bestemte tidsintervaller tildeles ny retning og hastighed. Hastighed og retning kan udregnes ud fra følgende ligninger:

$$s_n = \alpha s_{n-1} + (1-\alpha)s + \sqrt{1-\alpha^2} s_{x_{n-1}} \tag{2.4}$$

$$d_n = \alpha d_{n-1} + (1-\alpha)d + \sqrt{1-\alpha^2} d_{x_{n-1}} \tag{2.5}$$

Hvor s_n er den nye hastighed og d_n er den nye retning ved tidsintervallet n. α er en variabel i intervallet [0,1], der varierer tilfældigheden. s og d er konstanter, der repræsenterer middelværdien af hastighed og retning for n->(uendelig). $s_{x_{n-1}}$ og $d_{x_{n-1}}$ er tilfældige variabler fra en normalfordeling.

Fordele

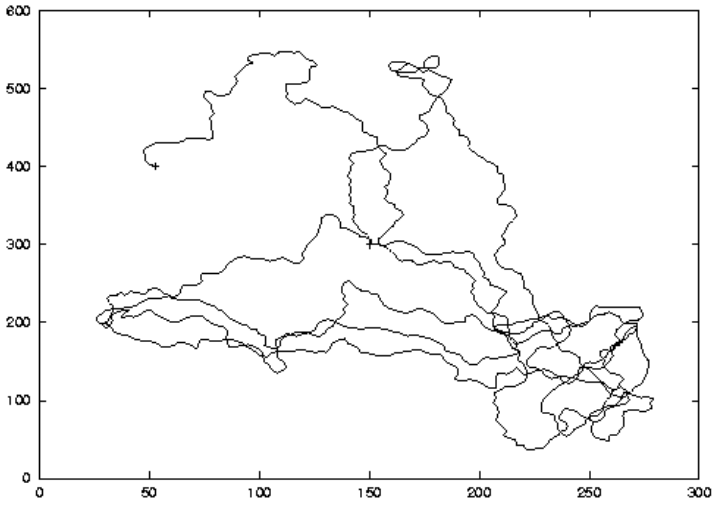
- Vil før eller siden bevæge sig over hele området.
- Man undgår mange af de skarpe sving og pludselige stop, som vil forekomme i f.eks. Random Direction.

Ulempe

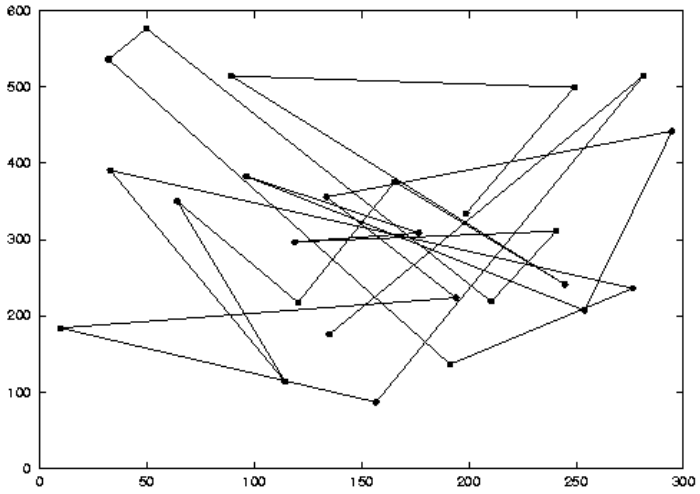
- Den bevæger sig over det samme område flere gange.
- Tiden, det tager at dække arealet, er uforudsigelig.

2.4.4 Random Waypoint

Denne algoritme kræver at området, der skal køres på , er kendt. På denne måde kan der vælges en tilfældig destination inden for området. Derudover vælges en tilfældig hastighed mellem en foruddefineret max- og minimumhastighed. Hver gang man når en destination, holdes en foruddefineret pause, og der vælges tilfældigt en ny destination inden for området.



Figur 2.5 *Gauss-Markov-modellen*



Figur 2.6 *Random waypoint-modellen*

Fordele

- Bevæger sig i hele området.

Ulemper

- Bevæger sig over det samme område flere gange.
- Tiden, det tager at dække arealet, er uforudsigelig.

2.4.5 Genetic Algorithm

Genetic Algorithm er en løsningsmetode til Travelling Salesman Problemet. Hvis man skal finde den bedste rute mellem en række punkter, skal man ikke have ret mange punkter, før det reelt bliver umuligt at beregne alle ruter og finde den korteste. Genetic Algorithm går ud på, at man tager en række tilfældige ruter mellem punkterne, vælger de to korteste og kombinerer dem til forhåbentligt to kortere ruter. Derefter erstatter man de to længste ruter med de nye. Dette gentages, indtil man ikke længere opnår en forkortelse af ruten. Den rute man når frem til, vil være meget tæt på den bedste rute.

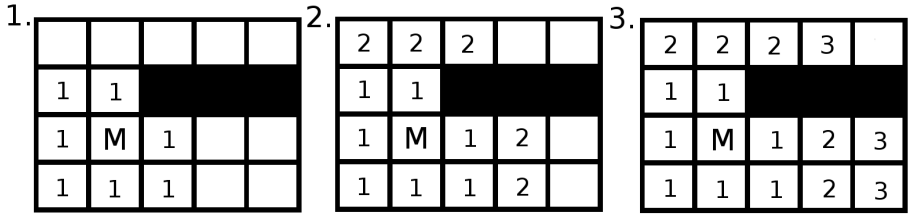
- Minimalt areal, der bliver overkørt flere gange.

Ulemper

- Området skal være kendt.
- Der skal fordeles punkter i området.
- Mange udregninger

2.4.6 Distance Transform Path Planning

Denne algoritme opdeler området, der skal køres på, i celler, se figur 2.9. Alle celler tildeles en værdi efter afstanden fra den celle, der angives som slutfeltet. Værdierne tildeles ud fra målcellen, ved at tildele nabocellerne værdien 1. Nabocellerne, der ikke allerede har tildelt en værdi, til hver af disse celler, tildeles værdien 2. Sådan fortsættes der til alle celler har tildelt en værdi, som vist på figur 2.7.



Figur 2.7 *Her ses hvordan tildelingen af værdier til de forskellige celler foregår.*

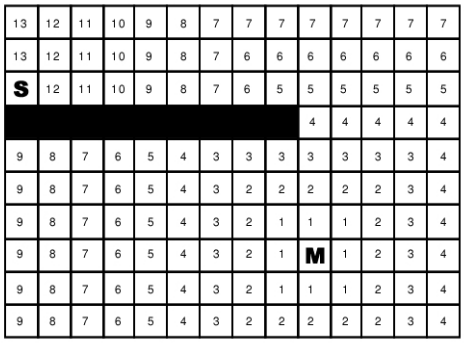
Derudover holdes der øje med hvilke celler, der er blevet dækket. Til at starte med sættes alle celler derfor til at være "ubesøgte". Der tjekkes nu på nabocellerne til den nuværende position, hvilke der har den højeste afstandsværdi og om de er markeret som "besøgt". Nabocellen, der opfylder disse kriterier vil så blive nuværende celle.

Fordele

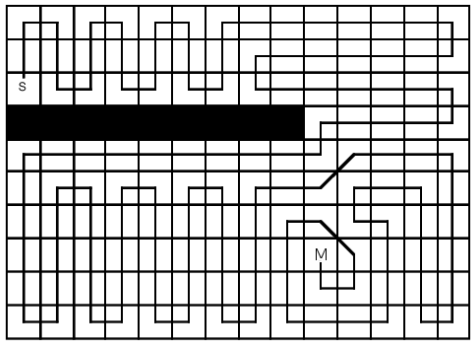
- Kører så vidt muligt kun over det samme område én gange.
- Er struktureret.
- Er effektiv.
- Start og slut position kan defineres.

Ulemper

- Skal have kendskab til området.
- Kræver udregnet kort med afstand til cellerne.



Figur 2.8 *Distance transform path planning. Her er tildelte værdier vist. Utilgængelige felter er sorte. Algoritmen tager udgangspunkt i feltet "S" og stopper i feltet "M".*



Figur 2.9 *Distance transform path planning. Her er den beregnede rute vist.*

2.4.7 Delkonklusion

I dette afsnit har vi kigget på forskellige rutealgoritmer, der kan benyttes til dækning af et givent område. Blandt de rutealgoritmer vi har beskrevet, er der flere der kan benyttes til forbedring af dt eksisterende produkter. I det initierende problem har vi stillet spørgsmålet:

- Hvilke rutealgoritmer kan bruges til at slå græsplænen i et system, hvor plæneklipperen i så vidt muligt omfang ikke kører over det samme sted mere end højest nødvendigt?

Visse af de tilfældighedsbaserede algoritmer kan måske klippe mere effektivt, end de nuværende anvendte algoritmer, men for at være sikre på, at vi så vidt muligt undgår, at klippe det samme sted flere gange, skal der benyttes en planlægningsalgoritme. Dvs. at de eneste to algoritmer vi kan bruge er Genetic Algorithm og Distance Transform Path Planning. Genetic Algorithm er den algoritme, der formentlig vil give den korteste rute, men Distance Transform Path Planning(DTPP) er en del simplere at implementerem og kræver langt færre beregninger. DTPP vil derfor være den bedste algoritme at anvende til vores løsning.

2.5 Matlab test af Random Direction

Som led i vores Matlabkursus og for at understøtte vores påstand om, at Random Direction algoritmen er ineffektiv med hensyn til dækning af et areal ved hjælp af den kortest mulige rute, har vi lavet et matlab-script[4] til at teste algoritmen i et kvadratisk område. Derudover er scriptet en prototype til det program, der vil blive det senere produkt.

Kvadratets areal er 500x500 pixels og stregens bredde er sat til 11 pixels. Da det er en tilfældighedsbaseret algoritme, har vi kørt modellen 10 gange for at få et udtryk for, hvor lang en rute Random Direction gennemsnitligt skal bruge, for at dække området.

[4]Se appendix D og Matlab.zip bilag F.1

Forsøg	Resultat
1	2,6888e5
2	2,9801e5
3	2,8736e5
4	3,7770e5
5	3,5752e5
6	3,5456e5
7	2,8693e5
8	3,8708e5
9	2,9521e5
10	2,6927e5

Tabel 2.2 Vores Matlabscripts udregninger af rutelængderne, af de 10 beregnede ruter i pixel. Det er gjort ved at summere længden af de vektorer, ruten består af.

Ud fra de 10 forsøgsresultater i tabel 2.2 beregnes gennemsnitslængden til 3,18252e5 pixel. Det maksimalt dækkede område ved denne længde vil være

$$\text{Rutelængde} \cdot \text{Klippebredde} \tag{2.6}$$

altså

$$3,18252e5 \cdot 11 = 3,500772e6 \tag{2.7}$$

hvor størrelsen på det område, der skal dækkes, kun er

$$500 \cdot 500 = 2,5e5 \tag{2.8}$$

Ved at sætte disse to tal op mod hinanden, finder vi ud af, hvor effektiv Random Direction algoritmen er, mht. at dække et areal med den kortest mulige rute.

$$\frac{3,500772e6}{2,5e5} = 14,003088 \tag{2.9}$$

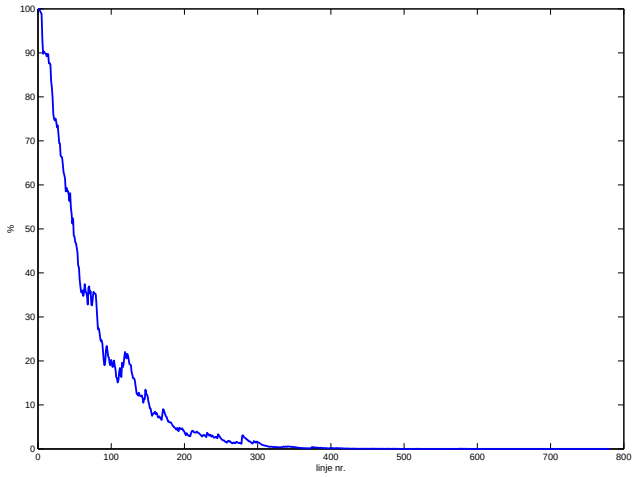
Det vil sige at vores rutesimulering af Random Direction tilbagelægger et areal, der er over 14 gange så stort som det areal, der bliver dækket. På figur 2.10 ses, hvor hurtigt effektiviteten af rutens områdedækning falder. En teoretisk optimal rute ville give en graf, der konstant havde en y-værdi på 100%.

2.6 Positionering

En del af de undersøgte rutealgoritmer kræver, at den mobile enhed kan stedbestemmes. I dette kapitel kigger vi på forskellige eksisterende og fremtidige muligheder for positionering. Formålet er, at undersøge om der findes en positioneringsmetode, som er velegnet til positionering af autonome plæneklipper.

2.6.1 Global Positioning System (GPS)

GPS[7] er et satellitbaseret navigationssystem, som består af i alt 24 satellitter, som kredser omkring jorden i en højde af 20.200 kilometer, i 6 forskellige kredsløb. Satellitterne kredser omkring jorden to gange i døgnet, og er beregnet således, at der i et hvilket som helst givent område på kloden altid vil være mindst 4 satellitter i direkte sigte. Hver satellit broadcaster konstant tid fra dens 4 sammensatte atomure i en broadcast-pakke. Pakken indeholder endvidere informationer om satellittens kredsløb om jorden og dens position i kredsløbet, satellittens status og et estimat af de andres aktive satellitter



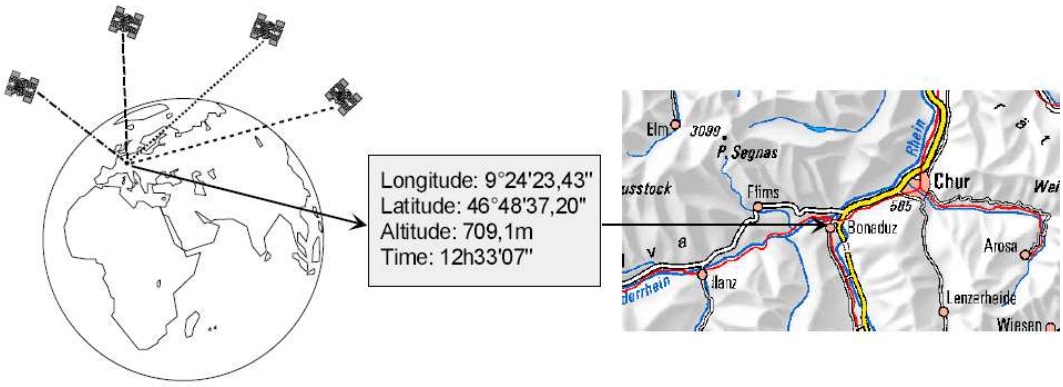
Figur 2.10 *Her ses en effektivitetskurve for Random Direction algoritmen i en Matlab model. På x-aksen vises antallet af linier i modellen. På y-aksen vises, hvor stor en procentdel af den enkelte linie, der dækker et udækket område.*

i nærheden. Estimatet giver en mulighed til GPS-modtageren; kun at søge efter satellitter, som er i sigte.

GPS-modtageren beregner deres position i breddegrad, længdegrad, højde over havets overflade og den præcise tid ud fra triangulering, ved at måle afstanden imellem mindst 4 satellitter. Når GPS-modtageren har beregnet afstanden mellem de 4 satellitter, kan modtageren beregne sin absolutte position, ved at kende satellitternes placering i kredsløbet og ved at lave en fiktiv kugle omkring hver satellit med afstanden som radius.

Præcisionen på GPS er, efter at U.S. military lukkede deres kunstige desorienteringssystem (SA, Selective Availability), i teorien under 10 meter. Systemet desorienterede positionen op til 100 meter, hvilket medførte at fjendtligtsindede ikke kunne bruge GPS-positionering til positionering af missiler.[8]

Det danske Farvandsvæsen oplyser på deres hjemmeside, at deres positionsnøjagtigheden i praksis er bedre end 20 m 95% af tiden.[10]



Figur 2.11 *Billedet viser placeringen af en GPS-modtager[8] Side. 9*

Årsagen til positioneringsfejl med GPS

- Selvom alle satellitter har 4 antomurer, vil en fejl på blot 10 ns, give en fejl på 3 m.
- Generelt er satellitternes position i kredsløbet, kun kendt inden for 1 til 5 meter.
- Signalet fra satelitten til modtageren bliver langsommere, når signalet bevæger sig igennem Ionosfæren.

Fordele

- Har en mulig præcision på under 20 m 95% af tiden
- Virker over alt i verden.
- Eksisterer allerede

Ulemper

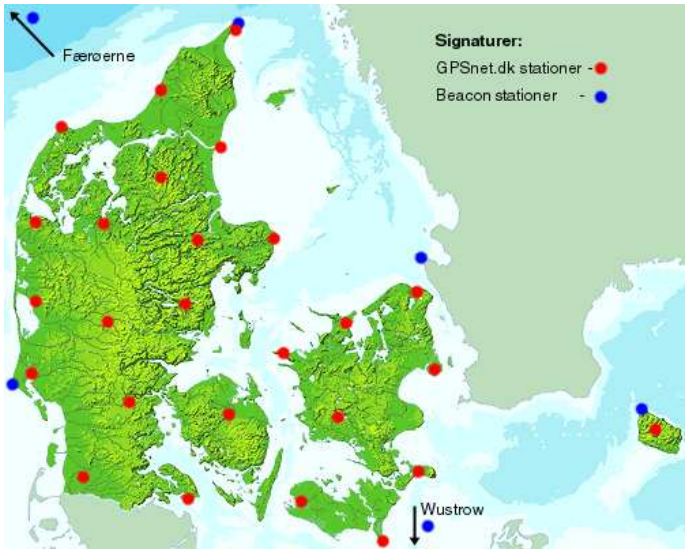
- Kræver at himlen kan ses.

2.6.2 Differential Global Positioning System (DGPS)

[7] DGPS er baseret på samme teknik som GPS, men for at forbedre præcisionen, er det blevet udbygget med fast placerede GPS modtagere i landskabet. Modtagerne beregner fejltolerancen mellem GPS signalet position og den egentlige position, som er kendt. Herefter bliver differencen broadcasted, som et FM signal som mobile GPS enheder modtager, og påregner deres relative position.

I Danmark er der DGPS stationer i Hammerodde, Blåvandshuk og Skagen, der udsender korrektion via FM. Disse stationer har farvandsvæsenet ansvaret for. Farvandsvæsenet oplyser på deres hjemmeside, at deres målinger med DGPS har en præcision på under 2 meter.[10]

Endvidere er der 26 Internet baserede reference stationer, hvilke giver en præcision på mellem 20 - 30 cm inden for en radius af 50km af reference station. [11]



Figur 2.12 *Billede viser DGPS reference stationer i Danmark, blå prikker er FM baserede, mens røde er Internet baserede [11].*

Fordel

- Har en mulig præcision på mellem 20-30 cm.
- Eksister allerede.

Ulemper

- Kræver at himlen kan ses.
- Kræver at der skal laves opkobling til DGPS netværket, enten via FM eller via internettet.

2.6.3 Galileo

[9] Galileo er baseret på samme praktiske virkemåde som GPS. Begrundelsen for Galileo's eksistens er at GPS er styret af U.S. military, hvilket betyder at, U.S. military har den fulde magt over systemet, hvilket kan betyde at de kan sætte begrænsninger eller lukke for alt civil brug af GPS systemet i krise perioder, hvis de finder det nødvendigt. For at undgå disse omstændigheder har man valgt at opsætte et ekstra navigation system til civile formål. The European Union og European Space Agency opnåede i marts 2002 enighed om at finansiere projektet, som omfatter opsendelsen af 30 satellitter mellem 2006-2010. Det forventes at være klar til brug for civile i 2010.

Der vil være 4 forskellige service muligheder til rådighed med Galileo.

- **Open Service (OS)** Open Service vil blive en gratis service for alle. Signalet vil blive udsendt i to forskellige bånd, hvis man vælger at benytte begge bånd vil man opnå en præcision på under 4 m horisontal og under 8 m vertikal, hvorimod hvis man kun bruger den ene af de to bånd, vil man opnå en præcision på under 15 m horisontal og under 35 vertikal.
- **Commercial Service (CS)** Commercial Service af Galileo vil blive en krypteret version, som giver en præcision på under 1 meter mod betaling. Endvidere er det muligt at forbedre præcisionen ved at benytte difference stationer på samme måde som med DGPS, positionen vil da opnå en præcision på under 10 cm.
- **Public Regulated Service (PRS) og Safety of Life Service (SoL)** Begge disse services vil være krypteret og vil give en præcision på samme højde som Open Service. Disse services er udviklet med specifikt henblik på at gøre dem mere hårdføre overfor støj, som måtte være genereret for at ubrugeliggøre systemet, eller gør det upræcist. Formålet for disse services er tiltænkt sikkerhedsautoriteter og flykontrol.

Fordele

- Skulle give en præcision under 4 m horisontal og under 8 m vertikal. Envidere skulle det blive muligt at forbedre det til under 10 cm via differentiale beregninger.
- Vil kunne bruges over alt i verden.

Ulemper

- Kræver at himlen kan ses.
- Er ikke færdig udviklet.
- De oplyste præcision nøjagtigheder er ikke testet i praksis.

2.6.4 Odometri

Odometri[6] er den mest benyttede metode til navigation for mobile enheder. På kort afstand giver odometri en meget præcis position og er en meget billig positioneringsteknik. Grundprincippet ved odometri er, at beregne bevægelseshastigheden samt tiden, hvor bevægelsen blev udført. Derved kan man beregne sig frem til sin egentlige position, ved hjælp af afstanden man har bevæget sig samt startpositionen. Odometris væsenligste problem er præcitionens usikkerhed på lange afstande. Problemet er, at når først der er opstået en fejl, vil robotten fortsætte, og næste gang der igen opstår en fejl, vil fejlen blive en sum af begge fejl. Derfor vil afvigelsen blive væsentligt forstærket i forøgelsen af afstanden.

Afvigelserne er opdelt i to fejl kategorier.

Konstruktionsfejl

- Hjul størrelsen afviger fra det specificerede.
- Justeringsfejl.
- Usikkerhed på akselafstanden.
- Sensorfejl på hjulene.

Miljø betonede fejl

- Målings fejl pga. at overfladen er ujævn.
- Overfladen er glat.
- Hjulene skrider pga. hurtig acceleration.
- Modstand i systemet pga. slitage.

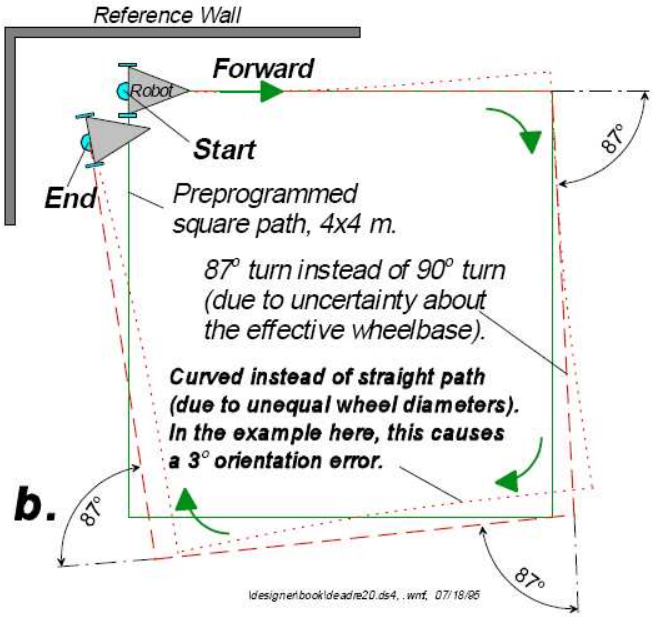
På figur 2.13 vises fejl afvigelsen på en firkantet rute på 4x4 meter. Billede 2.13 viser to forskellige konstruktions/slitagefejl. Den stiplede linje viser fejlsymptomerne ved afvigelse i akselafstanden. Hvis akselafvigelsen er større end det beregnede vil det resultere i, at robotten ikke får drejet det antal grader som ønsket. Hvorimod hvis akselafvigelsen er mindre end det beregnede, vil robotten dreje skarpere end det ønskede. Den stiplede linje viser fejlsymptomerne af forskellige hjul størrelser, hvilket ofte vil kunne opstå pga. slitage, fejlen betyder at robotten vil dreje til den side, hvor hjulet er mindst. Begrundelsen er, at hvis robotten skal køre ligeud, vil hjulene have den samme rotationshastighed, mens rotations længden vil være forskellig fra hinanden pga. forskellen.

Fordele

- Er en billig positioneringsløsning.
- Er meget præcis på korte afstande.
- Virker indendørs.

Ulemper

- Bliver upræcis på lange afstande.
- Kræver stabil kontakt til kørefladen.
- Fejl eskalerer/hænger ved.



Figur 2.13 *Billede viser odemetri afvigelser ved hendholdsvis akselafvigelse og hjulstørrelse afvigelse [6] Side. 133*

2.6.5 Delkonklusion

I dette kapitel har vi kigget på forskellige positionerings metoder generelt. Før vi kan bruge nogle af positioneringsmetoderne, er vi nødt til at overveje, hvilke krav der stilles, for at benytte positioneringsmetoden på en autonom plæneklipper. Overfladen, en autonom plæneklipper kører på, vil ofte være en ujævn græsoverflade. Derfor er positioneringsmetoden nødt til, at kunne abstrahere fra overfladen den kører på. Desuden skal man overveje, hvor præcis præcisionen egentlig skal være, for at være acceptabelt. Det mest optimale vil være, at plæneklipperen kunne ramme kanten præcist hver gang. Af de positioneringsmetoder der er blevet gennemgået i kapitlet, er der ingen, der har en præcision, der vil kunne opnå dette krav. Derfor vil det kunne blive et krav, at man er nødt til at overskride kanten(fra tidligere omgange) med den afstand, der svarer til positioneringensusikkerhed, for at være sikker på, at arealet er dækket. For at give et eksempel, hvis positioneringen er uprækis med 10 cm, vil det kræve at en plæneklipper på 50 cm, vil være nødt til at dække kanten med 20 cm. De første 10 cm er for at dække den igangværende kørsels usikkerhed og de andre 10 cm er for at dække den usikkerhed i hvor kanten egentlig er. Resultatet vil betyde, at klippebredden kun vil være 30 cm.

Af de positioneringsmuligheder vi har undersøgt, vil den mest oplagte positionering være Galileo, da den skulle tilbyde den bedste nøjagtige position. Problemerne ved Galileo er, at det ikke er færdig udviklet endnu, og at den præcision som de forskriver, skulle være mulig, ikke er testet fuldt ud i praksis. En anden oplagt mulighed vil være GPS, men de forsøgsresultater som GPSnet.dk[11] har beskrevet med DGPS, giver kun en præcision positions nøjagtighed på mellem 20-30 cm, hvilket vil betyde at en autonom plæneklipper på 50 cm, kun vil give en effektivitet på 20-30 cm. klippe bredde.

En forbedringsmulighed vi har undersøge var, selv at sætte en reference station op i nærheden, og derved få en bedre korrektion. Desværre har vores leden været forgæves, men vi har fundet en mulighed, at bruge ”Virtual Reference Station (VRS)”, som er er en løsning som GPSnet.dk[12] tilbyder. Virkemåden er den samme som DGPS, forskellen er blot at VRS[12] beregner en virtuel reference station i nærheden af ens position, ved at lave en beregning mellem alle de fysiske reference stationer. Forsøg på landinspektør-uddannelsen på Aalborg universitet 2004 viste, at positions nøjagtigheden var 9 mm i planen og 16 mm i højden[12]. Derfor vil GPS med VRS på nuværende tidspunkt, være en oplagt mulighed for positionering af autonome plæneklipper. Eneste problem vil kunne være anskaffelsesprisen, og prisen for at benytte VRS teknikken ved GPSnet.dk[11].

2.7 Produktvision

Vores produktvision er baseret på erfaringen fra problemanalysen om, at de eksisterende autonome plæneklippere på markedet ikke er effektive i forholdet mellem det areal de klipper, og den afstand de tilbagelægger, da de kører over det samme sted flere gange, og har problemer med at komme ind i smalle områder. Derfor er vores vision, at forbedre effektiviteten, så tid samt ressourcer kan spares. I kapitel 2.2 har vi kigget på de eksisterende produkter, og undersøgt deres klippede areal over tid og deres rutealgoritmer. Ved hjælp af kapitel 2.4 kan vi konkludere, at de brugte rutealgoritmer ikke er effektive, da de kører over det samme punkt flere gange end nødvendigt. Derfor er første punkt, at vi vil prøve at undersøge muligheden for, at bruge en anden rutealgoritme(Distance Transform Path Planning) , og hvilke krav der vil være til ændringer af hardware. Vi vælger algoritmen Distance Transform Path Planning, da den gør det muligt, at dække et givet område på en struktureret måde. Vi har beskrevet produktvisionen ud fra, hvordan vi forstiller os en overordnet løsning på vores initierende problem vil virke. Væsentlige krav er så vidt muligt, at kunne undgå at køre over det samme område mere end én gang. For at dette er muligt, er man nødt til at vide, hvor man allerede har været. Derudover kræves det, at den autonome plæneklippers position hele tiden er kendt.

Nedenfor har vi beskrevet i punktform, hvordan vi forestiller os produktvisionens arbejdsrutiner udføres, for at blive mere effektive.

1. Efter den autonome plæneklipper er aktiveret, skal den først finde kanten
2. Når kanten er fundet, kortlægges plænen ved først at køre hele kanten af plænen rundt (Samtidig slår den kanten rundt)
3. Når den er færdig, beregner den et rutekort ud fra den afsluttede placering på kortet
4. Herefter gennemkører den rutekortet

2.8 Konklusion

På baggrund af vores problemanalyse, kan vi konkludere, at der er forbedringsmuligheder i det område, vi lagde ud med, at ville beskæftige os med i vores initierende problem. Så vidt vi har kunnet finde ud af, kører de nuværende produkter på markedet ud fra en tilfældighedsbaseret algoritme lig Random Direction algoritmen. Vi har derfor testet Random Direction algoritmen i en Matlabmodel. Hvor vi viste, at det er en ineffektiv algoritme at køre efter. Vi har desuden undersøgt og beskrevet forskellige alternative algoritmer, der kunne bruges til at dække et område, og er nået frem til, at Distance Transform Path Planning er den mest optimale til vores løsning.

2.9 Problemformulering

I problemanalysen fandt vi ud af, at der er mulighed for forbedring af de nuværende autonome plæneklipperes rutealgoritme. Vi vil derfor i resten af projektet fokusere på, hvordan vi kan lave et program, der kan simulere kørsel med henholdsvis planlægningsalgoritmen DTPP og Random Direction algoritmen i vilkårligt formede områder.
Derefter vil vi sammenligne de to, for at finde og dokumentere svaret på spørgsmålene:

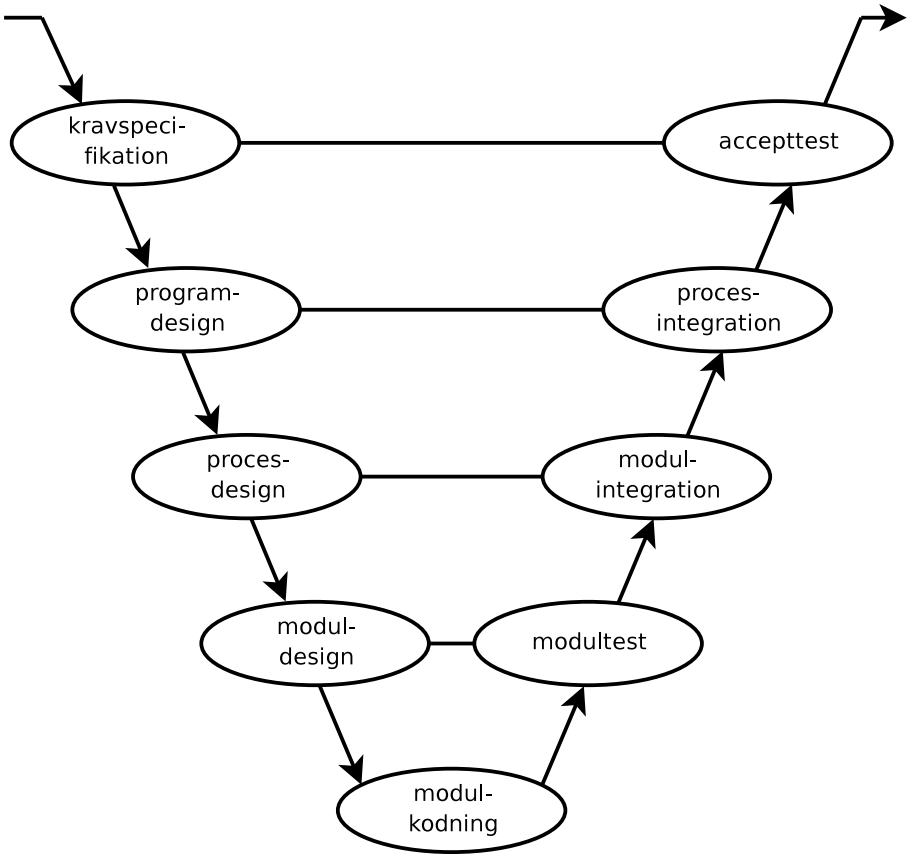
- Hvilken rutealgoritme er mest effektiv Random Direction eller Distance Transform Path Planning?
- I hvilke situationer er Random Direction bedre end Distance Transform Path Planning?
- I hvilke situationer er det modsatte tilfældet?

Kapitel 3

SPU modellen

3.1 SPU udviklingsmodellen

I vores projekt er løsningen blevet udarbejdet efter SPU[1] udviklingsmodellen, som vi under P2 har haft forelæsninger i. SPU udviklingsmodellen har til formål, at gøre udviklingen af software mere struktureret, ved at lægge nogle faste rammer for, hvordan udviklingen af softwaren skal foregå, i form af en række faser. Vi vil her beskrive disse faser, dog uden at komme ind på argumentationen for hvert enkelt fase i SPU.



Figur 3.1 *SPU modellen.*

[1] Struktureret ProgramUdvikling

- **Kravspecifikation**
  Første del af SPU-modellen er kravspecifikationen. Formålet med denne er, at fastsætte utvetydige og testbare krav til den software, man vil udvikle, samt specificere en accepttest, for til slut at kunne undersøge om kravspecifikationen er overholdt.
- **Programdesign**
  Derefter udarbejdes programdesignet, og procesintegrations-specifikationen. Formålet er, at specificere de eksterne grænseflader for programmet, og opdele programmet i et antal parallelle processer.
- **Procesdesign**
  Procesdesignfasen går ud på, at designe den enkelte proces, og få processen opdelt i moduler, samt fastsætte specifikationerne for disse moduler.
- **Moduldesign**
  I moduldesignfasen bestemmes, hvordan modulerne skal udføre deres funktioner og specifikationer for, hvordan disse moduler skal testes påbegyndes.
- **Modulkodning**
  Efter disse indledende trin kommer modulkodningen, og selve programmeringen.
- **Modultest**
  I modultestfasen bliver modultest specifikationerne færdiggjort, og der bliver konstrueret modultest programmer. Derefter testes modulerne og gøres klar til modulintegration, og der laves en testrapport.
- **Modulintegration**
  De færdige moduler bliver integreret trinvis. Derefter gøres processerne klar til procesintegration, og der laves en testrapport.
- **Procesintegration**
  Processerne bliver integreret trinvis, det færdige produkt afleveres til accepttest, og der laves en testrapport.
- **Accepttest**
  Til slut udføres accepttesten, som er specificeret i kravspecifikationen, for at sikre, at programmet lever op til de krav, der er stillet i kravspecifikationen. Der laves en accepttest rapport, og en eventuel brugervejledning gøres færdig.

Kapitel 4

Kravspecifikation

4.1 Indledning

4.1.1 Formål

Formålet med udarbejdelsen af kravspecifikationen er, at kunne give en veldokumenteret beskrivelse af programmets omfang, og give en beskrivelse af, hvad programmet vil indeholde af begrænsninger og resultater. Kravspecifikationen er blevet udarbejdet på baggrund af undervisningen i SPU[1] på forårssemesteret 2006 for Datateknik på Aalborg Universitet, samt bogen Struktureret Programudvikling [5]. Metoden, vi har brugt til udarbejdelse af kravspecifikationen, er ved at bruge bogens metoder kontinuert igennem hele udarbejdelsesfasen. Formålet med udviklingen af produktet er, at vise et bedre løsningsforslag, til at dække et areal med mindst mulig ressourcespild. Det vil vi gøre ved at bruge en rutealgoritme, som i højere grad end de nuværende brugte rutealgoritmer, er i stand til at undgå, at køre over de samme steder gentagne gange i simuleringsområdet.

Produktet vil blive udarbejdet med henblik på, at kunne simulere en autonom plæneklippers rute på en græsplæne ved hjælp af en software applikation. Vi vil udelukkende bruge softwaresimulation til at teste rutealgoritmerne med.

4.1.2 Læsevejledning

For at få det bedste udbytte af kravspecifikationen, anbefales det, at man læser kravspecifikationen igennem kronologisk. Vi anbefaler endvidere, at man har kendskab til UML, da vi bruger det til udformningen af forskellige diagrammer.

4.2 Generel beskrivelse

4.2.1 Systembeskrivelse

Brugeren af programmet skal kunne indtegne et to-dimensionalt sammenhængende område grafisk eller indtaste koordinaterne for kantpunkterne i meter i den rækkefølge, de skal forbindes. Den angivne kant må ikke krydse sig selv. Hvis brugeren alligevel angiver en kant, der krydser sig selv, skal programmet give brugeren en fejlmeddelelse. Programmet skal kunne optegne området vha. vektorer, og derefter beregne og indtegne en rute på baggrund af en rutealgoritme. Når ruten er simuleret, skal programmet kunne angive noget statistik over, hvor effektiv dækningen af området har været. Med effektiv dækning menes der, hvor stort et areal der er blevet dækket[2], i forhold til hvor stort arealet af området er. Der skal være mulighed for at vælge imellem Distance Transform Path Planning og Random Direction algoritmerne på en liste. Der skal være mulighed for at tilføje flere algoritmer som ekstra moduler. Da vi skal teste to fundamentalt forskellige algoritmer, er pro-

[1] Struktureret Programudvikling

[2] for forståelsens skyld kan ”dækket areal” tilnærmelsesvist beskrives som rutelængde x klippebredde

grammet nødt til at have algoritmespecifikke funktioner.

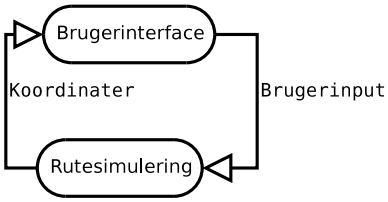
Ved Distance Transform Path Planning algoritmen[3] skal programmet tegne et grid med værdier for hver af de enkelte felter over det indtastede område. Grid'et skal kunne skaleres i forhold til området, så alle områdestørrelser kan simuleres.

Ved Random Direction algoritmen[4] skal programmet logge det areal, der bliver dækket, og stoppe simuleringen, når arealet er dækket. Det vil desuden være nødvendigt at gentage simuleringer af tilfældige rutealgoritmer, for at få et bedre billede af deres gennemsnitlige effektivitet.

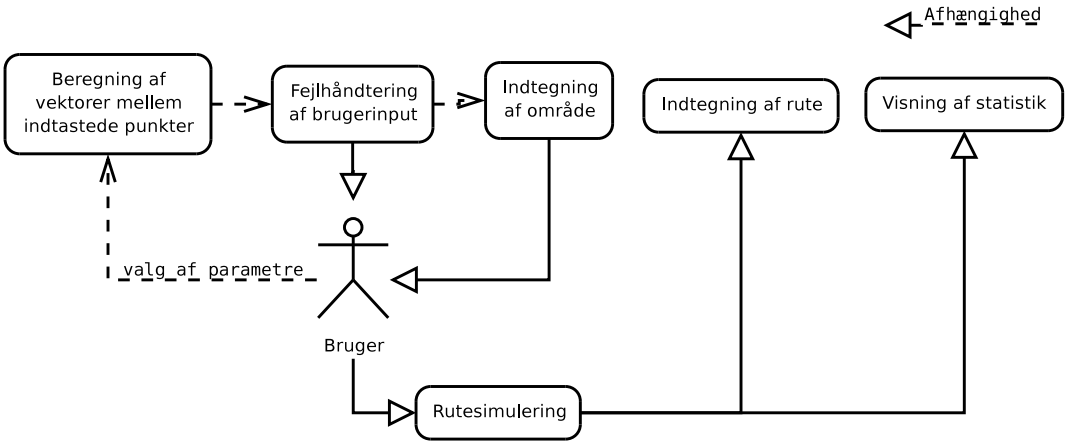
4.2.2 Programmets funktion

Programmets processer:

- Ruteberegning og -simulering
- Brugerinterface (visualisering af rute)

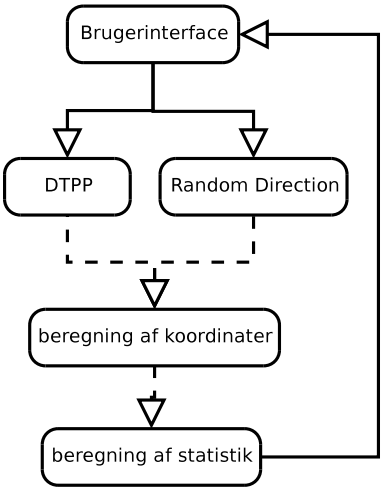


Figur 4.1 *Programmets processer med dataflow. Brugerinterfacet sender de informationer, brugeren angiver, til rutesimuleringen, der så planlægger og derigennem simulerer ruten. Efter endt simulering sender rutesimuleringen de beregnede koordinater til ruten tilbage til brugerinterfacet. Brugerinterfacet bruger herefter koordinaterne til at visualisere ruten.*



Figur 4.2 *Dataflow-Diagram over funktioner i brugerinterfaceprocessen. I denne del visualiseres programmets data og brugerinput håndteres. Når brugeren er tilfreds med sine indtastede parametre, sendes disse videre til ruteplanlægningsdelen. Dataene, der sendes tilbage fra ruteplanlægningsdelen, vises til brugeren.*

[3] Se 2.4.6
[4] Se 2.4.2



Figur 4.3 *Dataflow-diagram over moduler i ruteplanlægningsprocessen. Parametrene fra brugerinterfacedelen behandles vha. den valgte rutealgoritme.*

4.2.3 Programmets begrænsninger

Under programmets begrænsninger vil vi specificere, hvad systemet ikke kan. Formålet med klart og tydeligt at specificere begrænsningerne er på et tidligt tidspunkt, at afklare hvad det er nærliggende at tro, systemet kan.

- Det skal ikke være muligt, at definere runde hjørner i områdets kant. Runde hjørner vil være opbygget af mindre lige linjestykker. Det vil være op til brugeren, at placere koordinaterne til linjestykkerne.
- Kun ét område kan defineres. Dvs. at hvert punkt, der angives, forbindes med det forrige. Derved kan der kun angives ét område. Der kan altså ikke defineres forhindringer inden for området.
- Der skal ikke være mulighed, for at ændre punkterne i programmet, når de først er indtastet og gemt. Hvis de skal ændres, skal det gøres manuelt i den gemte fil.

4.2.4 Programmets fremtid

Programmet vil kunne udvides med:

- yderligere rutealgoritmer
- Visuel simulering af dækningen af området

4.2.5 Brugerprofil

Brugerne af programmet er medlemmerne af gruppe ED2A302 på Aalborg Universitet, da programmet laves specifikt til sammenligning af algoritmer i dette projekt.

4.2.6 Krav til udviklingsforløb

Til programudviklingen benyttes SPU. Programmeringssproget er C.

4.2.7 Forudsætninger

For at afvikle vores program tilfredsstillende, skal der benyttes en x86-baseret computer på minimum 1 GHz og med minimum 256 MB RAM. På computeren skal der bruges Linux(kernel 2.6) som styresystem. Desuden skal Imagemagick-pakken[5], samt en teksteditor være installeret. Programmet køres gennem c fortolkeren CH[6].

4.3 De specifikke krav

4.3.1 Definitioner

Effektivitet	Områdets areal delt med tilbagelagt areal.
Tilbagelagt areal	Defineres som rutelængde · klippebredde

4.3.2 Funktionelle krav

I dette afsnit vil vi formulere de forskellige funktioner i vores program. Funktionerne er beskrevet efter en uformel metode, baseret på en standardopdeling for hvert enkelt punkt.

4.3.2.1 Gem punkt

Indledning	Når brugeren har indtastet et punkt, han er tilfreds med, gemmes dette vha. denne funktion.
Input	I denne funktion er input, de koordinater brugeren indtaster. Koordinaterne må ikke ligge udenfor det angivne område, som brugeren kan sætte punkter i.
Funktion	Når brugeren trykker ”Gem”, gemmes det angivne punkt i en fil, samt i et dobbelt array, såfremt det bliver godkendt af ”Fejlhåndtering”. Første gang et punkt gemmes, skal der angives et filnavn til den fil, punkterne gemmes i. Hvis ”Fejlhåndtering” afviser koordinatet, gives en fejlmeddelelse til brugeren.
Output	Arrayet sendes videre til ”Tegn område”.

4.3.2.2 Load

Indledning	Her har brugeren mulighed for, at hente data fra en fil med tidligere angivne punkter.
Input	Tekstfil og den tilhørende billedfil.
Funktion	Funktionen henter punkter fra en tekstfil og gemmer dem i et dobbelt array.
Output	Arrayet sendes videre til ”Tegn område”.

[5] www.imagemagick.org
[6] http://www.softintegration.com/

4.3.2.3 Fejlhåndtering

Indledning	Funktionen sørger for at vektorerne kun afgrænser ét område.
Input	Her bruges koordinaterne som brugeren indtaster til input.
Funktion	Funktionen sammenligner alle skæringspunkter med de angivne punkter. Hvis der er flere skæringspunkter end angivne punkter, og/eller der er mere end to vektorer i ét skæringspunkt, meldes fejl til brugeren, og punktet gemmes ikke.
Output	Funktionen godkender eller afviser koordinatet til ”Gem punkt”, afhængigt af om koordinatet er gyldigt.

4.3.2.4 Tegn

Indledning	Ud fra punkterne tegner programmet det angivne område, og præsenterer det for brugeren. Der tegnes en ny streg, hver gang et nyt punkt tilføjes. Ved ruteindtegning tegnes der ligeledes en streg, hver gang et nyt punkt tilføjes.
Input	Her bruges arrayet fra ”Gem punkt” eller ”Load”som input, samt billedet der blev oprettet i forbindelse med forrige punkt.
Funktion	Programmet tegner området grafisk ud fra de angivne punkter i arrayet.
Output	Det tegnede område gemmes som en billedfil. Billedfilen nanvgives ud fra den tilhørende tekstfils navn.

4.3.2.5 Visning af statistik

Indledning	Formålet med funktionen er, at vise statistikken for effektivitet, rutelængde og den estimerede kørselstid over den simulerede rutealgoritme.
Input	Her bruges datastrukturen fra ”Beregning af statistik” som input.
Funktion	At vise statistikken til brugeren i et vindue.
Output	

4.3.2.6 DTPP

Indledning	Formålet med denne funktion er, at implementere rutealoritmen Distance Transform Path Planning i programmet.
Input	Her bruges startpunkt og områdets koordinater som input.
Funktion	At beregne rutens koordinater indenfor området ud fra Distance Transform Path Planning.
Output	Den beregnede rutes koordinater.

4.3.2.7 Random Direction

Indledning	Formålet med denne funktion er, at implementere rutealoritmen Random Direction i programmet.
Input	Her bruges startpunkt og områdets koordinater til input.
Funktion	At beregne rutens koordinater indenfor området ud fra Random Direction.
Output	Den beregnede rutes koordinater.

4.3.2.8 Beregning af statistik

Indledning	Formålet med denne funktion er, at udregne statistik over ruten.
Input	Her bruges rutens koordinater og hastighedsparametrene, der er angivet af brugeren bruges til input.
Funktion	At udregne effektiviteten og længden af ruten. Ud fra hastighedsparametrene udregnes, hvor lang tid ruten ville tage at gennemføre.
Output	Struct med statistikberegningerne.

4.4 Eksterne grænsefladekrav

4.4.1 Brugergrænseflade

Der skal være et felt til indtastning af kantkoordinater, samt felter til indtastning af hastighed og vinkelhastighed til brug for statistikudregning. Der skal vises et område med den indtegnede rute.

4.4.2 Software-grænseflade

Programmet skal afvikles under et Linux styresystem(Kernel 2.6)

4.5 Krav til programmets ydelse

Fra sidste punkt er gemt, og simuleringen påbegyndes, må der højst gå 30 sekunder, før simuleringen er færdig, og en rute fremvises på det angivne område.

4.6 Kvalitetsfaktorer

Pålidelighed	Programmet skal mindst have en oppetid på hvad der svarer til et gennemløb. Desuden må der ikke være fejl i beregningerne i programmet.
Vedligeholdelsesvenlighed	Er ikke vigtigt da programmet ikke skal bruges efter dette projekt.
Udvidelsesvenlighed	Er meget vigtigt da vi, hvis tiden tillader det, vil udvide programmet med et visualiseringsmodul, til at vise kørslen af den beregnede rute. Derudover kan det komme på tale at skulle simulere flere algoritmer, hvilket naturligvis også vil kræve en udvidelse.
Brugervenlighed	Ikke særlig vigtigt da programmet ikke skal bruges af personer udenfor gruppen.
Integritet	Det er særdeles vigtigt at programmet undervejs gemmer data i en fil, da angivne områder ellers ikke kan genskabes nøjagtigt, med mindre koordinater er nedskrevet.

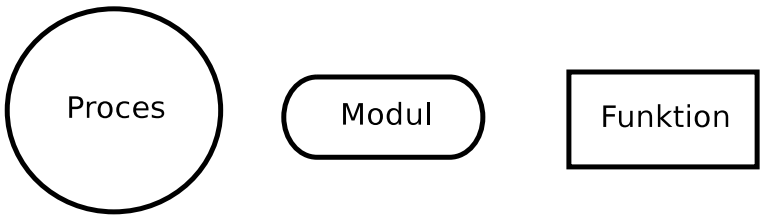
Kapitel 5

Design

Designkapitlet er udarbejdet på baggrund af afsnittet om design i [5], med henblik på at opfylde kravene i kravspecifikationen, kap. 4.

5.1 Programdesign

Programdesignet er blevet udarbejdet, med henblik på at kunne definere, hvilke overordnede moduler programmet består af. Forklaringen af symbolerne i diagrammerne i dette kapitel, kan ses på figur designbilledforklaring.



Figur 5.1 *Billedforklaring til samtlige diagrammer i design dokumentet*

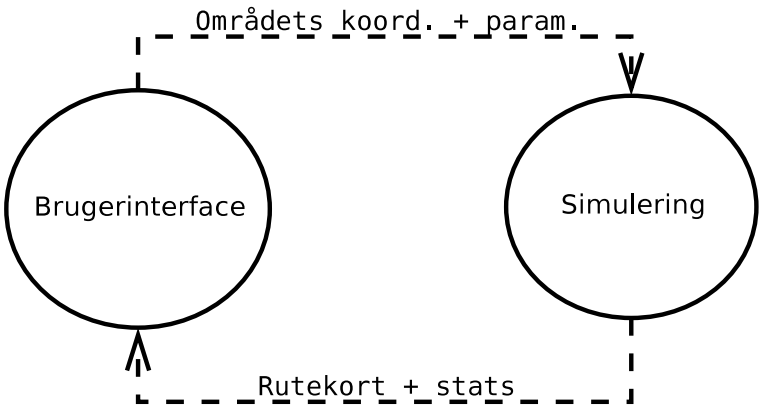
Vi har valgt, at designe programmet vha. top-down-metoden. Denne metode starter fra toppen med design af processer, og udarbejder derefter systematisk programmet ned igennem de forskellige moduler.

Proces inddelingen og grænsefladen mellem processerne er, som vist på figur 5.2 rutesimulering og brugerinterface.

Rutesimuleringsprocessen implementerer af hensyn til kravene i kap. 4.2.1, den valgte rutealgoritme. Der kan simuleres to forskellige algoritmer: Distance transform path planning (se kap. 2.4.6) og Random direction (se kap. 2.4.2. Af hensyn til kravene i kap. 4.2.1 er det i dette modul, de statistiske beregninger bliver foretaget. Det være sig effektiviteten af algoritmen, længden af ruten i meter og hvor lang tid det vil tage, at køre ruten, ud fra de parametre brugeren har indtastet i brugerinterfacet. Brugerinterfaceprocessen håndterer alt interaktion med brugeren. En filhåndteringsdel tillader brugeren at hente koordinater til et omraade ind i programmet. Denne fil oprettes og redigeres med en ekstern editor. Der er dog også mulighed for at gemme et allerede indlæst område og den dertil hørende beregnede statistik. Det er her brugeren indtaster de parametre, hvorudfra nogle af de statistiske beregninger og simuleringen skal udføres. Disse parametre kan ses i følgende tabel:

5.2 Procesdesign

Hver proces er defineret således, at de fungerer som en selvstændig enhed, som kan sepereres og fungere i selvstændige tests. For at gøre dette muligt, benytter processerne sig i såvidt mulig omfang

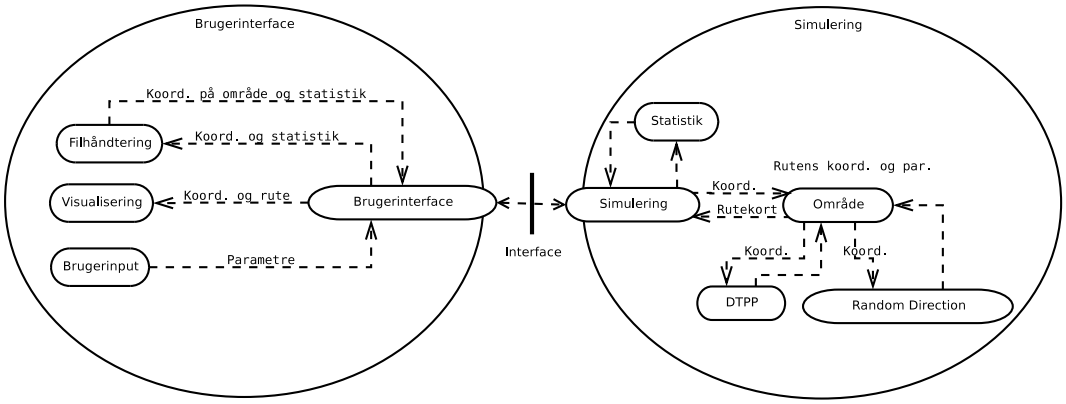


Figur 5.2 *Diagram over processerne og deres interface.*

Parameter	Enhed
Klippebredde	cm
Opløsning af det generede billede	pixels pr. m
Kørehastigheden	m pr. sek.
Vinkelhastigheden i svingene	° Grader pr. sek.

Tabel 5.1 Tabel over parametrene, som brugeren kan definere. Klippebredden, opløsningen og bredden af området gør det muligt, at opfylde kravet i kap. 4.2.1 om skalerbarhed.

kun af lokale variabler.



Figur 5.3 *Diagram over moduler og dataflowet mellem dem*

5.2.1 Brugerinterface

Brugerinterface moduler er alle sammen moduler, som enten giver brugeren interaktionsmuligheder i programmet eller præsenterer data for brugeren.

- **Brugerinterface modulet** fungerer som interface til simuleringsprocessen.
- **Visualisering** styrer alt, der har med grafisk fremstilling at gøre. Dette er både den grafiske brugergrænse, f.eks. dialogbokse, og fremvisning af data og billeder.
- **Brugerinput** håndterer alle input fra brugeren og events[1].
- **Filhåndtering** håndterer åbning og lukning filer, herunder formatering af data i filerne.

5.2.2 Simulering

Simuleringsprocessen implementerer den valgte algoritme og genererer en rute, som gemmes som koordinater og rutekort i form af et bitmap billede. Dette gøres på baggrund af det område, brugeren har indtastet. Herudover beregnes statistik på baggrund af den genererede rute og de parametre, brugeren har indtastet.

- **Simulering**-modulet fungerer som interface til brugerinterface processen.
- **Område** modulet indeholder områdets koordinater, der videresendes til den valgte rutealgoritme, der skal simuleres. Endvidere holder dette modul øje med, hvor meget af området, der er dækket vha. analyse på et billede, der også genereres her. Grænsefladerne mellem dette modul og henholdsvis DTPP- og RD-modulerne skal derudover være identiske af hensyn til kravet i kap. 4.2.1 om, at det skal være muligt, at tilføje flere algoritmer. Dette er nemmest, hvis der er et fast defineret interface.
- **DTPP**-modulet genererer en rute i form af koordinater ud fra Distance Transform Path Planning rutealgoritmen. Først skabes en matrix, hvor hvert felt i matrixen symboliserer en kvadratisk celle. Cellen har en bredde, der er defineret af brugeren i brugerinterfacet. Først markeres hvilke celler kanten af området gennemløber, det være start- og slutfeltet i matricen. Hver celle indenfor området bliver tildelt en værdi, som beskrevet i kap. 2.4.6. Herefter beregnes ruten ud fra algoritmen.
- **Random direction**-modulet genererer en rute, ved at vilkårligt vælge en ny retning mellem 0 og 180° i forhold til den kant, som den rammer. Derefter placeres ruten mellem disse vilkårlige punkter. Modulet køres indtil område modulet melder, at området er dækket.
- **Statistik** modulet beregner statistik af den simulerede rute af hensyn til kravet i kap. 4.2.1.

5.3 Moduldesign

Efter procesdesignet, vil vi i dette kapitel gennemgå hvert enkelt modul og dets funktioner. Når hvert enkelt moduls funktionalitet er bestemt, opdeles modulerne i funktioner. Idéen i at opdele hver modul i funktioner er, at de nemt kan genbruges, og giver en bedre gennemskuelighed af programmet. Derudover gør det det nemmere at gennemteste hver funktion for sig.

Beskrivelserne af modulerne indeholder følgende punter: funktion og reference til kravspecifikationen. Input og output vil fremgå af pseudokoden, som er indrammet i en kasse.

[1] En funktion der startes af brugeren f.eks. ved klik på en knap

5.3.1 Brugerinterface

- **Funktion:** Modtager brugerinput og viser område, rute og statistik. Desuden tegnes området i dette modul.
- **Reference til kravspecifikation:** kap. 4.2.1 på side 31.

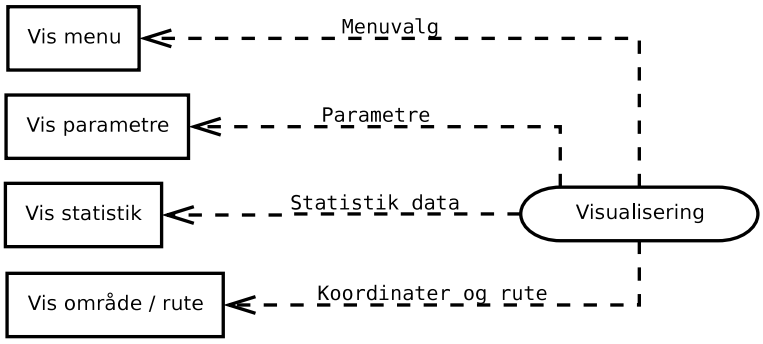
5.3.1.1 Brugerinterface-modul

```
1  main()
2  {
3    while()
4    {
5    visualisering(0) //Visualiserer menu mulighederne.
6    brugerinput(0, char &key) //Input for menu.
7
8    select case key
9    case a //Åbn fil med koordinater og statitistik.
10     filhaandtering(areal, statistik, parameter, 0)
11   case g //Gem fil med koordinater og statistik.
12     filhaandtering(areal, statistik, parameter, 1)
13   case s //Statistik.
14     visualisering(areal, parameter, statistik, 1)
15   case o //Vis område.
16     visualisering(areal, parameter, statistik, 2)
17   case c //Vis område og rute.
18     visualisering(areal, parameter, statistik, 3)
19   case p //Vis parameter.
20     {
21     while(key) != 'e' //indtil der bliver trykket e for exit.
22       {
23         visualisering(areal, parameter, statistik, 4)
24         brugerinput(1,&key) //Input for parameter
25         select case key
26         case k //set klippebredde
27         case c //set drejningstid
28         case h //set hastighed
29         case e //set start punkt
30         case o //set stop punkt
31         end select
32       }
33     }
34   case k //Aktiver simuleringsprocessen.
35     simulering_modul(areal, parameter, statistik)
36   end select
37   }
38 }
```

5.3.1.2 Visualisering

- **Funktion:** At fremstille den grafiske brugerflade inkl. fremvisning af område og rute.
- **Reference til kravspecifikation:** kap. 4.2.1 på side 31.

```
visualisering(areal *areal, parameter *parameter, statistik *statistik, int nr)
{
  select case nr.
  case 0
    vis_menu()
  case 1
    vis_statistik(statistik *statistik)
  case 2
    vis_omraade_og_rute{areal, 0)
  case 3
    vis_omraade_og_rute{areal, 1)
  case 4
    vis_parametre(parameter *parameter)
  end select
}
```



Figur 5.4 *Diagram over funktionerne i visualiseringsmodulet, som består af funktionerne "vis statistik" og "vis område/rute".*

1. **Vis menu** funktionen viser hvilke muligheder brugeren har.

```
Vis menu()
hvis der er åbnet en fil endnu:
  {
    vis menupunktet:
      åbn fil
      gem fil
      parameteropsætning
      hvis der ikke er valgt et start- og stoppunkt
      {
        giv mulighed for at køre simulation
      }
      afslut program
      vis omraade
      vis rute
      vis daekket omraade
      vis statistik
  }

  ellers vis kun punkterne "åbn fil" og "afslut".

return menuvalg;
}
```

2. **Vis parametermenu** funktionen viser de forskellige parametre, der kan ændres.

```
1 vis_parametermenu(parameter *parameter)
2 {
3 print alle parameter mulighederne ud samt værdierne.
4 }
```

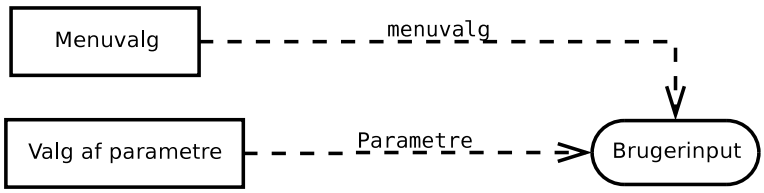
3. **Vis statistik** funktionen viser længden af ruten, dækket areal, areal af området og gennemløbstid.

```
1 vis_statistik(statistik *statistik)
2 {
3 print statistik dataene ud
4 }
```

4. **Vis område/rute** funktionen viser det definerede område på skærmen og ruten.

```
1  vis_omraade_og_rute{areal *areal, int nr)
2  {
3    if (nr == 0)
4    {
5      Generer billede af arealet, og derefter åbner det.
6    }
7    if (nr == 1)
8    {
9      Åbner det genererede billede med ruten fra simulerings delen.
10   }
11 }
```

5.3.1.3 Brugerinput



Figur 5.5 *Diagram over funktionerne i brugerinputmodulet*

- **Funktion:** Inputhåndtering/-validering.
- **Reference til kravspecifikation: kap. 4.2.1 på side 31**

```
1 brugerinput()
2 {
3 Hvis kommando hovedmenu
4 kør funktion menuvalg
5 Hvis kommando parametermenu
6 kør funktion parametervalg
7 }
```

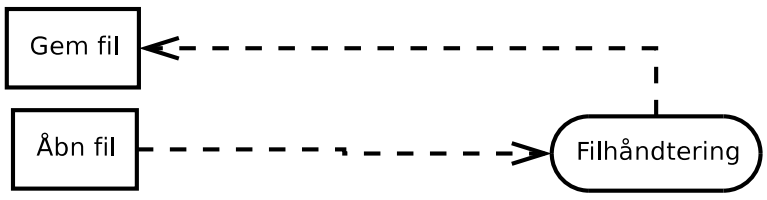
1. **Menuvalg** funktionen giver brugeren mulighed for at definere parametrene, som er vist i tabel 5.1.

```
menuvalg(areal *omraade){

  scanf();
  hvis der er åbnet en fil endnu:
  {
    giv mulighed for at vælge:
      åbn fil
      gem fil
      parameteropsætning
      hvis der ikke er valgt et start- og stoppunkt
      {
        giv mulighed for at køre simulation
      }
      afslut program
      vis omraade
      vis rute
      vis daekket omraade
      vis statistik
  }

  ellers giv kun mulighed for at åbbne en fil og afslutte programmet

return menuvalg;
}
```

2. **Valg af parameter** funktionen giver brugeren mulighed, for at definere parametrene, som er vist i tabel 5.1 på side 40.

```
parameteropsaetning(parameter *parameter, areal *omraade){
  scanf();

  giv mulighed for at:
  gå tilbage til hovedmenu
  indstille hastighed
  indstille bredde
  indstille marginbredde
  indstille vinkelhastighed
  indstille opløsning
  vælge algoritme
  indstille startpunkt (Tjek om punktet ligger inden i omraadet)
  indstille stoppunkt (Tjek om punktet ligger inden i omraadet)

returner menuvalg
}
```

5.3.1.4 Filhåndtering



Figur 5.6 *Diagram over funktionerne i filhåndteringsmodulet*

- **Funktion:** Filhåndtering giver brugeren mulighed for at gemme, og åbne filer med informationer om området, statistik og parametre. Ved åbning af en fil, må de indlæste koordinater ikke resultere i, at de vektorer der angiver simuleringsområdets kant, krydser hinanden. Ugyldige input vil resultere i en besked til visualisering om, at vise en besked om det til brugeren.

- **Reference til kravspecifikation:**kap. 4.3 på side 34.

```
filhaandtering(areal *areal, statistik *statistik, parameter *parameter, int nr)
{
  if (nr == 0)
  {
    aabn_fil(filnavn, fil, statistik,parameter,areal)
  }
  if (nr == 1)
  {
    gem_fil(filnavn, fil, statistik,parameter,areal)
  }
}
```

1. **Gem fil** funktionen gemmer koordinaterne på det tegnede område, samt statistik data for simuleringen.

```
gem_fil(char *filnavn, FILE* fil)
{
  fil = fopen(filnavn, "w");
  Gemmer derefter areal, parametre og statistik.
}
```

2. **Åbn fil** funktionen henter område koordinater fra en defineret fil samt , samt statistik data. Vektorerne verificeres ved, at beregne skæringspunktet mellem hver enkel vektor med følgende formler:

$$y_1 - \alpha = \beta \tag{5.1}$$

α og β beregnes for begge vektorer med formlen:

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1} x_1 \tag{5.2}$$

$$\left[\begin{array}{cc|c} 1 & \alpha_1 & \beta_1 \\ 1 & \alpha_2 & \beta_2 \end{array}\right] \tag{5.3}$$

$$\left[\begin{array}{cc|c} 1 & \alpha_1 & \beta_1 \\ 0 & \alpha_2 - \alpha_1 & \beta_2 - \beta_1 \end{array}\right] \tag{5.4}$$

$$\left[\begin{array}{cc|c} 1 & \alpha_1 & \beta_1 \\ 0 & 1 & \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \end{array}\right] \tag{5.5}$$

$$\left[\begin{array}{cc|c} 1 & 0 & \beta_1 - \alpha_1 \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \\ 0 & 1 & \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \end{array}\right] \tag{5.6}$$

$$\begin{bmatrix} \beta_1 - \alpha_1 \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \\ \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1} \end{bmatrix} = \begin{matrix} y \\ x \end{matrix} \tag{5.7}$$

```
aabn_fil(char *filnavn, FILE* fil)
{
  fil = fopen(filnavn, "r");
  Find skæringspunkt mellem alle vektorer to ad gangen.
  Hvis skæringspunktet ligger inden for begge vektors koordinater:
  returner fejlkode
  Henter derefter areal, parameter og statistik ud og filen via pointeren
  ''fil''
}
```

5.3.2 Simulering

5.3.2.1 Simulering-modul

- **Funktion:** Interface til Brugerinterface processen.
- **Reference til kravspecifikation:** kap. 4.2.1 på side 31.

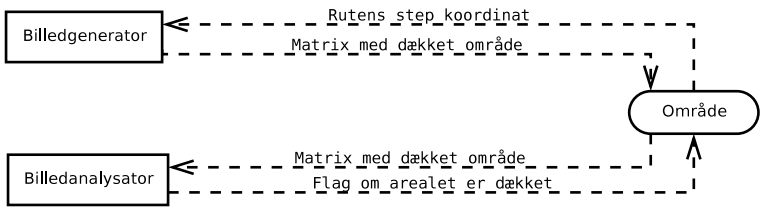
```
1 simulering(areal *areal, parameter *parameter, statistik *statistik)
2 {
3   omraade(areal,parameter);
4
5   statistik(areal, statistik, parameter);
6 )
```

5.3.2.2 Område

Dette modul indeholder vektorer, der afgrænser området, samt det dækkede areal.

- **Funktion:** Sender områdekoordinater videre til algoritmerne og holder øje med hvor meget af området, der er dækket.
- **Reference til kravspecifikation:** kap. 4.2.1 på side 31.

```
1  omraade(areal *areal, parameter *parameter)
2  {
3    if (parameter->rutealgoritme == 0) //DTPP
4    {
5      dtpp(areal, parameter); //Beregner ruten.
6      billedegenerator(areal, parameter, 0, 0); //Tegner arealet ind.
7      billedegenerator(areal, parameter, 2, 0); //Tegner hele ruten ind.
8    }
9
10   if (parameter->rutealgoritme == 1) //Random Direction
11   {
12   billedegenerator(areal,parameter,0,0) //Tegner arealet ind.
13
14     while(billedeanalyse()) //Indtil hele arealet er dækket.
15     {
16       nr = random_direction(areal, nr); /*Generer et ny rute step. Retuner
17                                           step nr.*/
18       billedegenerator(areal, parameter, 1 ,nr); //Tegner rute step ind.
19     }
20   }
21 }
```



Figur 5.7 *tekst*

1. **Billedegenerator** funktionen indtegner rutens step koordinater ind på et bitmap billede.

```
billedgenerator(areal *areal, parameter *parameter, int fu, int nr)
{
  if(fu ==0) //Tegn areal.
    {
      Tegn areal kanterne ind på tegningen med en sort streg,
      samt mal arealet udenom arealet sort.
    }
  if(fu ==1) //Tegn rute step ind.
    {
      Tegn det specificerede rute step ind.
      nr. definer hvilken rute step det er.
      parameter->bredde definer, hvor bred stregen skal tegnes med.
    }
  if(fu ==2) //Tegn hele ruten ind.
    {
    Tegner hele ruten ind.
    }
}
```

2. **Billedanalyse** funktionen beregner ud fra billede, om hele arealet er dækket.

```
int billedanalysator()
{
  Henter billede ind, og analyserer hvor mange felter, der ikke er farvet
  sort.
  Returner antal felter ikke sorte.
}
```

5.3.2.3 DTPP

Modulet DTPP implementerer Distance Transform Path Planning-rutealgoritmen og returnerer den genererede rute i en hægtet liste. DTPP modtager arealet i en hægtet liste som indeholder koordinater, som udgør kanten af det område, som rutealgoritmen skal bearbejde.

DTPP består af modulerne dtpp_find_border_cells, dtpp_calc_cell_values og dtpp_calc_route (se figur 5.10.

Modulet virker, ved først at overføre området til dtpp_find_border, der returnerer en matrix i form af et array med værdien -1 i de celler, hvor områdets afgrænsningerne går igennem. Herefter beregnes værdierne af de celler, der ligger indenfor cellerne med værdien -1. Dette gøres vha. modulet dtpp_calc_cell_values. Til sidst beregner den en rute igennem matrixen vha. rutealgoritmen, som beskrevet i kap. 2.4.6.

Klippebredden for DTPP algoritmen er nødt til at være større end bredden på cellerne, da der ellers kan opstå problemer med uklippede hjørner i cellerne ved skrå rutestykker, i situationer som illustreret på figur 5.8. Vi har derfor udregnet en margin der skal tilføjes cellebredden for at få den optimale klippebredde, som er illustreret på 5.9.

Den optimale margin er udregnet på følgende måde:

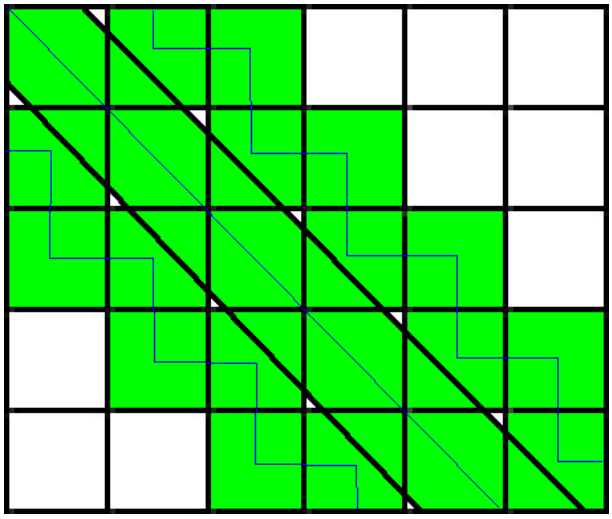
$$a = fracc - c \cdot cos(45^\circ)2 \tag{5.8}$$

$$k \cdot c \cdot cos(45^\circ) + k \cdot c = a \tag{5.9}$$

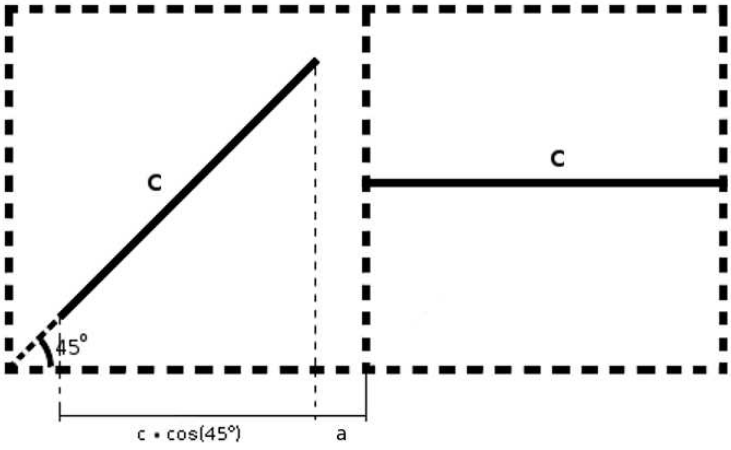
$$k = frac1 - cos(45^\circ)2(cos(45^\circ) + 1) = 0,0858 = 8,58\% \tag{5.10}$$

hvor c = cellebredde/klippebredde og k = procentdel af klippebredde der skal lægges til klippebredden på hver side.

For at sikre os mod cellehjørner der ikke bliver dækket, skal vi altså have en ekstra margin på 8,58 % af cellebredden tilføjet på hver side af vores klippebredde eller ialt 17,16 % af cellebredden, der skal lægges til.



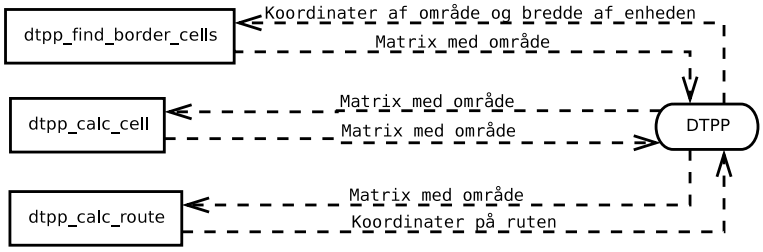
Figur 5.8 *Udsnit af område med beregnet rute (angivet med blå linier), hvor klippebredden er lig cellebredden. Der vil i disse situationer opstå problemer med uklippede hjørner af cellerne.*



Figur 5.9 *Forklaring af marginudregning*

- **Funktion:** Simulerer en rute ud fra Distance Transform Path Planning algoritmen.
- **Reference til kravspecifikation:** kap. 4.2.1 på side 31

```
1  dtpp(areal *omraade, parameter *parameter)
2  {
3  row = højden af arealet / bredde af enheden.
4  size = bredden af arealet / bredde af enheden.
5
6  Opret en matrix som et dobbelt array af arealet data[row][size].
7
8  //Marker arealet i matrixen ved at tilskrive ydergrænserne med -1.
9  dtpp_find_border_cells(omraade, row, size, data, st_parameter)
10
11 //Beregn matrixens DTPP værdier.
12 dtpp_calc_cell(row, size, data)
13
14 //Beregn ruten.
15 dtpp_calc_route(row, size, data, omraade)
16 }
```



Figur 5.10 *Billede viser DTPP modulet.*

1. **dtpp_find_border_cells** funktionen retunerer en matrix med arealet, der skal dækkes. (Se figur 5.11)

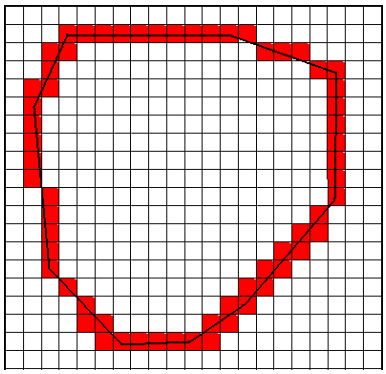
```
dtpp_find_border_cells(areal *omraade, int size, row,int data[row][size],
parameter *st_parameter)
{
  for(i=1;i>areal->antalkoord;i++)
  {
  Definerer hver område step som en vektor i forhold til matrixens
  størrelse, så hver celle er en kvadratisk celle, med enhedens bredde
  som celle bredde.

    //Beregner 100 punkter på hver vektor.
    for(q=0;100>q;++).
    {
      px = cast_int(areal->koordinater[i]->x + ((ax/100)*q))
      py = cast_int(areal->koordinater[i]->x + ((ay/100)*q))

    Til hver af punkterne bliver punktet, reflekteret over i matrixen,
    hver matrix celle som punktet er indenfor,
    bliver derefter tilskrevet værdien -1
    data[px][py] =-1
    }
  }
}
```

2. **dtpp_calc_cell** funktionen beregner cellernes DTPP værdier.

```
dtpp_calc_cell(int size, int row, in data[row][size])
While(zero)
  next++
  lock = 1
  While(lock)
  (
    lock = 0
     For(Antal felter vertikal)
    (
      For(Antal felter horisontal)
        (
        zero = 0
        if(felt værdi = next)
          (
    Sæt alle nabo felter, hvor der står nul i, til det markerede felts
    værdi+1.
          lock = 1
          )
        if(der står nul i det markerede felt)
          zero=1
        )
    )
  )
```

3. **dtpp_calc_route** funktionen beregner en rute der skal gennemkøres.

```
1  dtpp_calc_route(int row, int size, int data[row][size], areal *areal,
2  parameter *st_parameter)
3  {
4  Først sættes den føreste rute starts position til starts punktet
5  som er defineret i st_parameter->start
6    while(indtil der ikke er flere nabo celler, som er ledige.)
7    {
8      Ud fra starts punkt find nabo
9      celle med den største værdi, og som ikke er berørt.
10     Sæt derefter ruten step til nabo cellen.
11   }
12 }
```

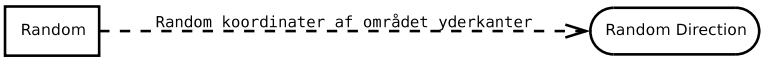


Figur 5.11 *Billede viser dtpp_matrix funktionens virkemåde. Felterne med den røde farve vil i matrixen blive skrevet med værdien -1*

5.3.2.4 Random Direction(RD)

Random Direction modtager arealet, som er en liste med vektorer, som omkredser arealet af det område, som rutealgoritmen skal bearbejde.

Modulet virker ved, at hvert step kalder den funktionen Random, som returnerer en ny tilfældig retning inden for vektorområdets kant. Modulet kaldes af område-modulet indtil området er dækket.



Figur 5.12 *Billede viser Random direction modulet*

- **Funktion:** Simulerer en rute ud fra Random Direction algoritmen.
- **Reference til kravspecifikation:** kap. 4.2.1

```
1  int random_direction(areal *areal,int nr)
2  {
3  double x, y;
4  random_direction_random(areal, x, y)
5  Kopier det næste rute punkt ind i ruten.
6
7  return nr.
8  }
```

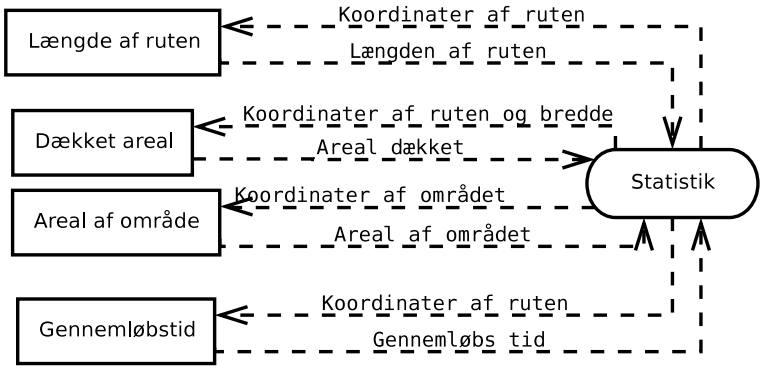
1. **Random** funktionen udvælger et tilfældigt tal mellem 0 og max-vinkel

```
random_direction_random(max-vinkel)
{
  vinkel = find tilfældigt tal mellem 0 og 1; gang det med 180.
returner vinkel
}
```

5.3.2.5 Statistik (Stat)

- **Funktion:** Beregner statistik ud fra den simulerede rute.
- **Reference til kravspecifikation:** kap. 4.2.1

```
statistik(areal *areal, statistik *statistik, parameter *parameter)
{
//Beregn længden af ruten.
laengde_af_ruten(areal, statistik)

//Beregning af areal kørt over.
daekket_areal(statistik, parameter)

//Beregning af området areal.
areal_af_omraadet(areal)

//Beregning af gennemløbs tid.
gennemlaebs_tid(areal, statistik, parameter)

//beregning af effektivitet
effektivitet(areal_af_området, dækket_areal)
}
```



Figur 5.13 *Billede viser "Beregning af statistik" modulet*

1. **Længden af ruten** funktionen returner længden af hele ruten, ved at beregne de samlede antals steps længde.

$$\sum_{k=1}^{\text{Antal Step}} \sqrt{(x_{k-1}-x_k)^2+(y_{k-1}-y_k)^2} = \text{Længden af ruten} \tag{5.11}$$

```
laengde_af_ruten(areal *areal, statistik *statistik)
{
for(antal af step - 1)
statistik->laeangde=+ pythagoras af afstanden mellem de to punkter.
}
```

2. **Dækket areal** funktionen beregner det totale areal enheden har kørt over, ved at beregne længden, der er kørt gange med bredde af enheden.

$$\text{Længden af ruten} \cdot \text{Bredde af enheden} = \text{Dækket areal} \tag{5.12}$$

```
1 daekket_areal(statistik *statistik, parameter *parameter)
2 {
3 statistik->daekket_areal = statistik->laengde * parameter->bredde
4 }
```

3. **Areal af området** funktionen beregner område størrelsen ud fra koordinaterne af området.

$$A = \frac{1}{2}(x1y2 - x2y1 + x2y3 - x3y2 + ... + xny1 - x1yn) \tag{5.13}$$

```
1 areal_af_omraadet(areal *areal)
2 {
3 Beregner arealet af området der skal dækkes
4 }
```

4. **Gennemløbs tid** funktionen beregner gennemløbs tiden for ruten. Måden tid bliver beregnet ud fra, er via længden af hele ruten, divideret med enhedens hastigheden i m/s. Samt summen af alle drejnings vinklerne, divideret med vinkelhastigheden.

$$\frac{\text{Længden af ruten}}{\text{hastighed}} + \frac{\sum \text{Drejningsvinkel}}{\text{vinkelhastighed}} = \text{Gennemløbstid} \tag{5.14}$$

```
1  gennemlaebs_tid( areal *areal, statistik *statistik, parameter *parameter)
2  {
3    for(antal step-1)
4    {
5      xf = data->rute[i+1]->y - data->rute[i]->y /*x koordinat på første
6                                                  vektor.*/
7      yf = data->rute[i+1]->x- data->rute[i]->x /*y koordinat på første
8                                                 vektor.*/
9      xa = data->rute[i+1]->c - data->rute[i+2]->x /*x koordinat på anden
10                                                   vektor.*/
11     ya = data->rute[i+1]->y- data->rute[i+2]->y /*y koordinat på anden
12                                                  vektor.*/
13     drejningsvinkel = drejningsvinkel + formel
14     i++
15   }
16 statistik->gennemloebstid = parameter->laengde/parameter->hastighed +
17 drejningsvinkel/parameter->vinkelhastighed
18 }
```

5.3.3 Hjælpemoduler

5.3.3.1 Linked List

Dette modul giver mulighed for, at benytte en datastruktur, baseret på en hægtet liste.

- Funktioner til at håndtere hægtet liste.

```
Point *newPoint(Coordinate x, Coordinate y)
{
  Point *p = malloc(sizeof(Point));
  p->x = x;
  p->y = y;
  return p;
}

List *newList(void)
{
  List *list = malloc(sizeof(List));
  list->count = 0;
  list->first = NULL;
  list->last = NULL;

  return list;
}

void insert(Point *point, List *list)
{
  Node *n = malloc(sizeof(Node));
  n->point = point;
  n->next = NULL;

  if (list->last == NULL)
    list->first = n;
  else
    list->last->next = n;

  list->last = n;
  list->count++;
}

void deleteFirst(List *list)
{
  Node *old;

  old = list->first;
  list->first = old->next;
  list->count--;

  free(old->point);
  free(old);
  old = NULL;
}
```

5.3.4 Datastrukturer

- Struct til parameter

```
struct parameter
{
  int hastighed
  float   bredde
  int vinkelhastighed
  int oploesning
  int rutealgoritme  //0=DTPP 1=Random Direction
}
```

- Struct til statistik

```
1 struct statistik
2 {
3   double laengde
4   double daekket_areal
5   double areal_af_omraadet
6   double gennemloebstid
7   double effektivitet
8 }
```

- Struct til areal koordinater

```
1  typedef double Coordinate;
2
3  typedef struct
4  {
5    Coordinate x;
6    Coordinate y;
7  } Point;
8
9  typedef struct
10 {
11   Point *point;
12   struct Node *next;
13 } Node;
14
15 typedef struct
16 {
17   Node *first;
18   Node *last;
19   int count;
20 } List;
21
22 struct areal
23 {
24   List *koordinater;
25   int antalkoord;
26   List *rute;
27   Point *start;
28   Point *stop;
29 }
```

-

- Struktur på datafilen med koordinater, parameter og statistik

```
1  0   0 //Følgende 5 felter er koordinater.
2  0   20 //Med hendholdsvis x og y.
3  10    20 //Hver koordinat angives på en linje for sig.
4  10    10 //x og y adskilles med en tabulator.
5  #slut på koordinater
6  hastighed 5
7  bredde 7
8  vinkelhastighed 50
9  rutealgoritme DTPP
10 laengde 10 //Længden af rute.
11 daekket_areal 300 //Dækket areal.
12 areal_af_omraade 100 //Område af arealet.
13 gennemloebstid 1200 //Gennemløbstid.
14 rutealgoritme      1 //Valgte rutealgoritme.
```

Kapitel 6

Test

For at sikre, at vores program fungerer optimalt og efter hensigten, har vi udarbejdet en række test til de forskellige processer og moduler. Der er dog i forbindelse med programmeringen gennemført flere tests af de forskellige moduler. Samtidig vil fejl i vores program ikke være kritiske, da de ikke vil være til risiko for andre mennesker, og programmet skal ikke ud til en kunde. Derudover er kompleksiteten af programmet begrænset, fordi det er begrænset hvor mange forskellige states, programmet kan være i. Derfor vil eventuelle fejl være nemme at rette, derudover er det begrænset hvor mange fejl, der kan være. På grund af dette er det ikke nødvendigt at teste så mange gange.

6.1 Modultest

Vi har valgt udelukkende, at bruge blackboxtests, til at teste modulerne i programmet med, da vi mener formålet med en whiteboxtest, er opfyldt gennem vores programmeringsfase. En whiteboxtest ville derfor tage for lang tid i forhold til, hvad vi kunne bruge den til. Allerede i programmeringsfasen er modulerne blevet testet adskillelige gange, for at sikre, at de kunne samarbejde. Vi har derfor valgt, kun at teste hvert modul og hver proces 5 gange i testfasen.

6.1.1 Filhåndtering

Åbn fil Ved funktionen ”Åbn fil” skal der testes for korrekt genkendelse af gyldige og ugyldige indlæste punkter. Dette gøres ved, at indlæse et antal punkter, man på forhånd kender gyldigheden af. Ved gyldige punkter skal det forventede output fra modulet, være en linket liste med de indlæste punkter. Ved ugyldige koordinater skal modulet give en fejlkode.

1. **Input:** Fil med gyldige kendte koordinater. F.eks. (0;0) (0;46) (46;46) (46;0)
   Forventet output: Linket liste med koordinater
2. **Input:** Fil med ugyldige kendte koordinater F.eks (0;0) (0;46) (46;0) (46;46)
   Forventet output: Fejlmeddelelse

Gem fil ”Gem fil” funktionen testes ved, at indlæse teststatistikdata i en struct på samme form som statistikdata output fra brugerinterfacemodulet, og derefter se om den gemmer dem korrekt i en fil.

- **Input:** Struct med teststatistikdata, som er forskellig fra nul, således at det er nemt at se om de rigtige tal er gemt.
  Forventet output: Fil med statistik

6.1.2 Visualisering

For at teste visualiseringsmodulet, indlæses koordinater og rute, hvorefter modulet køres. Alle gyldige menu-kommandoer indtastes og derefter ses, om menuen præsenteres rigtigt. Det skal være muligt, at få vist menu, parametre, statistik eller billede af området med rute.

1. **Input:** Menu-kommando. Denne er defineret som værdien 1.
   Forventet output: Menuvisning på skærm
2. **Input:** Parametervalg-kommando. Denne er defineret som værdien 7.
   Forventet output: Parametermenuvisning på skærm
3. **Input:** Statistik-kommando. Denne er defineret som værdien 9
   Forventet output: Statistikvisning på skærm
4. **Input:** Billed-kommando. Disse er defineret som 5, 6 og 11, alt efter hvilket billede, der skal vises.
   Forventet output: Billedvisning på skærm

6.1.3 Brugerinput

I dette modul skal brugeren, kunne indtaste de nødvendige parametre, der skal sendes videre i en struct. Derudover skal brugeren kunne vælge i en menu.

1. **Input:** Brugerens indtastning af menuvalg; pointer til struct parametre
   Forventet output: Den korrekte menu-kommando
2. **Input:** Brugerens indtastning af parametre. Der indtastes forskellige værdier til parametrene, således at det er nemt at se, om værdierne er blevet placeret rigtigt.
   Forventet output: Struct med parametre

6.1.4 Statistik

Indlæs rutekoordinater til en rute, man på forhånd har udregnet statistikken til.

- **Input:** Structs med hhv. testrute, testområde og testparametre. Testruten behøver ikke bestå af mere end 5 punkter, for at gøre det nemmere at regne statistikken efter manuelt.
  Forventet output: Struct med statistik. Tallene skal svare til de på forhånd beregnede tal

6.1.5 Område

Der indlæses først et områdes koordinater fra en testdriver og derefter en rutes koordinater løbende. Disse er på forhånd konstrueret til, at dække det angivne område. Derefter ses om modulet stopper rutesimuleringen, når området er dækket. Desuden testes om den, generer et korrekt billede på baggrund af rute- og områdekoordinater.

1. **Input:** Parametre, Område- og rutekoordinater. Parameteren bredde sættes til f.eks. 2 m. En rute indtegnes manuelt i et zig-zag-mønster, således at området med sikkerhed bliver dækket. Har man f.eks. et område med koordinaterne: (0;0) (0;10) (10;10) (10;0), en bredde på 2m, kan en rute f.eks. være: (0;0) (0;10) (2;0) (2;10) (4;0) (4;10) (6;0) (6;10) (8;0) (8;10) (10;0) (10;10), efterfulgt af tilfældige koordinater. De tilfældige koordinater burde ikke blive tegnet, fordi området gerne skulle være dækket. Rutekoordinater sendes sekvensielt fra en teststub.
   Forventet output: Stopsignal
2. **Input:** Område- og rutekoordinater. De ovenfor nævnte koordinater sammenlignes med det billede der bliver genereret.
   Forventet output: Korrekt billede af område med rute

6.1.6 DTPP

Først indlæses et områdes koordinater fra en testdriver, og derefter beregner modulet et antal rutestykker. De beregnede rutestykker skal ligge indenfor det angivne område.

- **Input:** Områdekoordinater
  Forventet output: Rutekoordinater inden for området

6.1.7 Random Direction

Først indlæses et områdes koordinater fra en testdriver, og derefter beregner modulet et antal rutestykker. De beregnede punkter for ruten skal ligge på de vektorer, der udgør området.

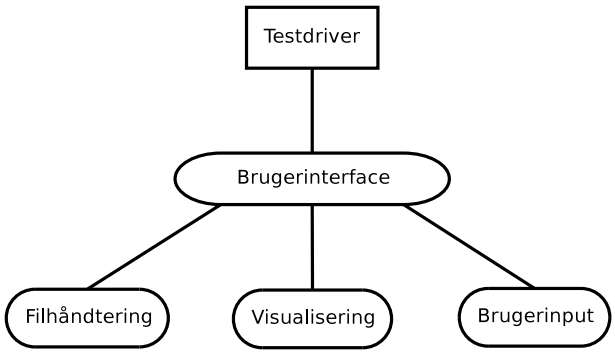
- **Input:** Områdekoordinater
  Forventet output: Rutekoordinater på områdekant

6.2 Modulintegration

Under Modulintegration testes procesdesignet. Modulerne i de forskellige processer skal testes sammen.

6.2.1 Brugerinterface

Da de enkelte moduler allerede er testet i 6.1, er det eneste, der skal testes her, integrationen mellem Filhåndtering-, Visualisering- og Brugerinputmodulerne, og Brugerinterfacemodulet, som det kan ses på figur 6.1.

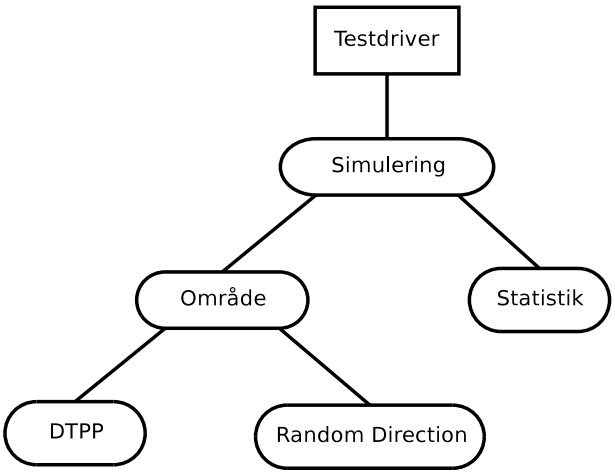


Figur 6.1 *Hierakisk opstilling af modulerne i Brugerinterface processen. Der testes ud fra bottom-up-metoden.*

1. **Input:** Bruger indtastninger
   Forventet output: Områdekoordinater og parametre i structs
2. **Input:** Structs med rutekort og statistik
   Forventet output: Visning af rutekort og statistik

6.2.2 Simulering

Først testes DTPP og Random Direction modulerne med område modulet. Derefter tilføjes Simulering og Statistik, som angivet på figur 6.2.



Figur 6.2 *Hierakisk opstilling af modulerne i Simulerings processen. Der testes ud fra bottom-up-metoden.*

- **Input:** Structs fra Brugerinterface med områdekoordinater og parametre
  Forventet output: Rutekort og statistik i structs

6.3 Procesintegration

Her testes integrationen mellem de enkelte processer. I vores program er der dog kun henholdsvis Brugerinterface- og Simuleringsprocesserne.

- **Input:** Brugerindtastninger (områdekoordinater og kørselsparametre)
  Forventet output: Billede af det angivne område med beregnet rute samt statistik

6.4 Accepttest

For at stress teste programmet, vil vi køre simuleringer med meget store områder og mange kantkoordinater. Der skal desuden testes for, hvor store områderne må være, når kravene til programmets ydelse skal opfyldes. Altså hvor stort et område der kan simuleres på den tid, der er angivet i afsnit 4.5 ved hjælp af en testmaskine, som beskrevet i afsnit 4.2.7.

For at teste om programmet opfylder kravene fra kravspecifikationen, vil vi undersøge, om følgende er overholdt:

- Brugeren skal kunne indtegne et to-dimensionalt sammenhængende område grafisk, eller ved indtastning af punkter.
- Hvis den angivne områdekant krydser sig selv, skal der gives en fejlmeddelelse.
- Programmet skal kunne optegne det angivne område og den beregnede rute.
- Programmet skal kunne vise statistik over dækningen af, det angivne område.
- Brugeren skal kunne vælge, at simulere kørsel med enten DTPP- eller RD algoritmen.
- Det skal være muligt at åbne en fil med områdekoordinater.
- Der skal være et felt til indtastning af kantkoordinater.
- Efter endt simulering, skal det angivne område med indtegnet beregnet rute vises.
- Der må højst gå 30 sek. fra simuleringen påbegyndes til området med ruten fremvises.

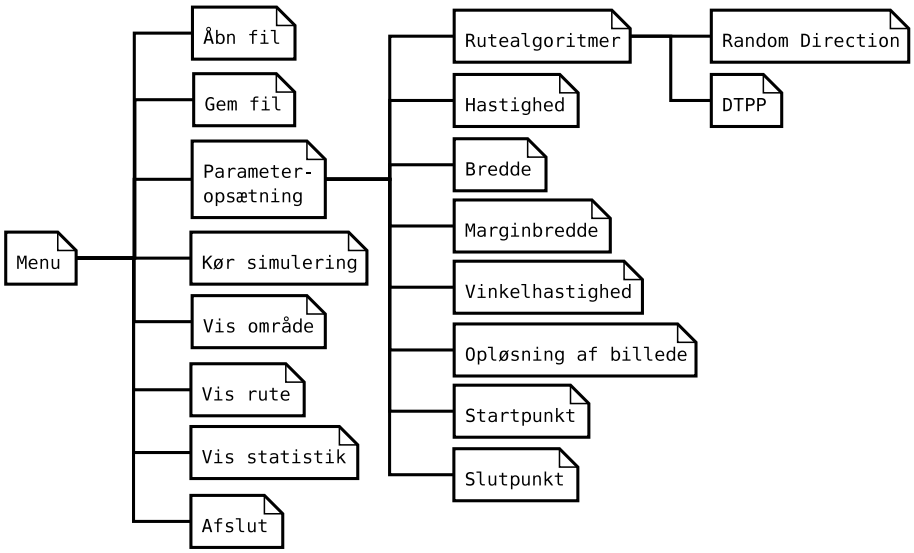
Kapitel 7

Implementation

Vi vil i dette kapitel give et overblik over programmet og uddybe de vigtigste funktioner, samt gennemføre tests af de forskellige programdele.

7.1 Programbeskrivelse

Vores program er lavet ved hjælp af programmeringssproget C. Det er menustyret via en konsolbaseret brugergrænseflade. Menuen er bygget op som vist på figur 7.1.



Figur 7.1 *Opbygning af programmets menusystem.*

For at opfylde vores krav om, at det skal være muligt, at implementere flere algoritmer i programmet[1], er der forskellige funktioner i statistik- og områdemodulerne afhængig af algoritmetypen.

For at bruge programmet, skal man oprette en datafil med områdekoordinater[2] på formen der er angivet i tabel 7.1. Det er muligt, at angive start- og slutpunkt i filen, men dette kan også gøres fra programmet. Efter områdekoordinaterne er læst ind i programmet, er det muligt, at vælge mellem de to algoritmer, indstille de forskellige parametre og derefter simulere kørsel i det angivne område. Efter endt simulering, kan man få vist den simulerede rute, og de dækkede områder. Det vil både før og efter simulering være muligt, at få vist det angivne område. Programmet er vedlagt på en CD i bilag F.1

[1] Se afsnit 4.2.1

[2] Se evt. afsnit 5.3.4 på side 55

X1	Y1
X2	Y2
X3	Y3
...	...
Xn	Yn

Tabel 7.1 Standard for hvordan koordinater skal skrives ind. I hver række skrives en X- og en Y-værdi, adskilt af en tabulator.

7.2 Testkørsel

7.2.1 Modultest

Filhåndtering

Krav	Kommentar	Tjek
5 linkede lister med korrekte koordinater ved 5 forskellige filer med gyldige koordinater		✓
5 fejlmeddelelser ved 5 forskellige filer med ugyldige koordinater		✓
5 filer med korrekt statistik ved 5 forskellige structs med statistikdata		✓

Visualisering

Krav	Kommentar	Tjek
5 korrekte menu-visninger ved 5 menu-kommandoer		✓
5 korrekte parametermenu-visninger ved 5 parametervalg-kommandoer		✓
5 korrekte statistik-visninger ved 5 statistik-kommandoer		✓
5 korrekte billed-visninger ved 5 billed-kommandoer		✓

Brugerinput

Krav	Kommentar	Tjek
5 korrekte menu-kommandoer ved 5 indtastninger af menuvalg		✓
5 structs med korrekte parametre ved 5 forskellige indtastninger af parametre		✓

Statistik

Krav	Kommentar	Tjek
5 structs med korrekt statistik ved 5 forskellige structs med rute og parametre		✓

Område

Krav	Kommentar	Tjek
5 forskellige sæt af område- og rutekoordinater		✓
5 korrekte billeder af område med rute ved 5 forskellige serier område- og rutekoordinater		✓

DTPP

Krav	Kommentar	Tjek
5 serier rutekoordinater inden for det angivne område ved 5 forskellige serier områdekoordinater		✓

Random Direction

Krav	Kommentar	Tjek
5 koordinater på det angivne områdes kant ved 5 forskellige serier områdekoordinater	Områdekoordinater skal angives med urets retning.	✓

7.2.2 Modulintegration

Brugerinterface

Krav	Kommentar	Tjek
5 structs med korrekte data ved 5 forskellige brugerindtastninger		✓
5 korrekte visninger af statistik og rutekort ved 5 forskellige structs med statistik og rutekort		✓

Simulering

Krav	Kommentar	Tjek
5 structs med korrekte data ved 5 forskellige structs med rutekort og statistik		✓

7.2.3 Procesintegration

Krav	Kommentar	Tjek
5 korrekte output ved 5 forskellige indtastninger		✓

7.2.4 Accepttest

Krav	Kommentar	Tjek
Indtegning/indtastning af punkter	Ved brug af ekstern editor	✓
Fejlmeddelelse ved krydsende kant		✓
Optegning af område og rute		✓
Statistikvisning		✓
Valg mellem DTPP- og RD algoritmen		✓
Løbende gemt fil med punkter	Fil med punkter skal oprettes på forhånd uden for programmet	X
Åbning af fil med områdekoordinater		✓
Indtastning af hastighed og vinkelhastighed		✓
Visning af område med beregnet rute		✓
Max 30 sek. fra simuleringen påbegyndes til visning af rute	Ingen af forsøgene, der er blevet udført i forsøgskapitlet, har taget længere tid end det angivne tidspunkt. Dog kan den maksimalt tilladte simuleringstid overskrides, hvis grafik canvaset bliver større end 2000x2000 pixel.	X

Stresstest
Programmet klarer uden problemer, at simulere kørsel i et område på 36000m^2 med DTPP algoritmen. Denne områdestørrelse er 20 gange større, end den maksimalt anbefalede områdestørrelse til de nuværende autonome plæneklippere. Det vil derfor være usandsynligt, at det bliver nødvendigt, at simulere større områder.

7.3 Forbedringsmuligheder

Når DTPP algoritmen lukker sig selv inde, finder programmet bare den nærmeste celle og springer til denne. Herefter fortsættes algoritmen. Det betyder at ruten i nogle områder vil gå i gennem kanten og uden for området. Dette kunne have været løst ved at generere nye kort, hvor celleværdierne tildeles med slutpunkt i alle de ledige en af gangen. DTPP algoritmen ville så kunne anvendes modificeret, således at enheden bevæger sig efter de laveste værdier omkring den i stedet for de højeste. Den celle, hvis rutekort giver den korteste rute, vil så være den næste celle. Enheden vil så skulle bevæge sig ad den den rute til cellen, som blev genereret, for at finde ud af at denne celle var den nærmeste.

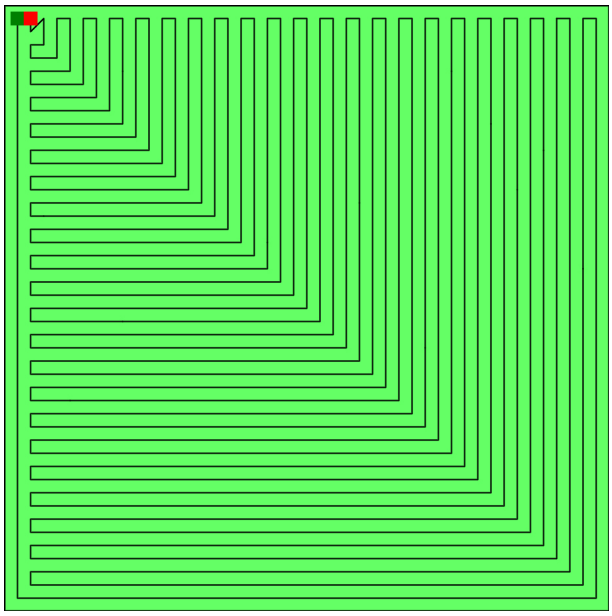
Kapitel 8

Forsøg

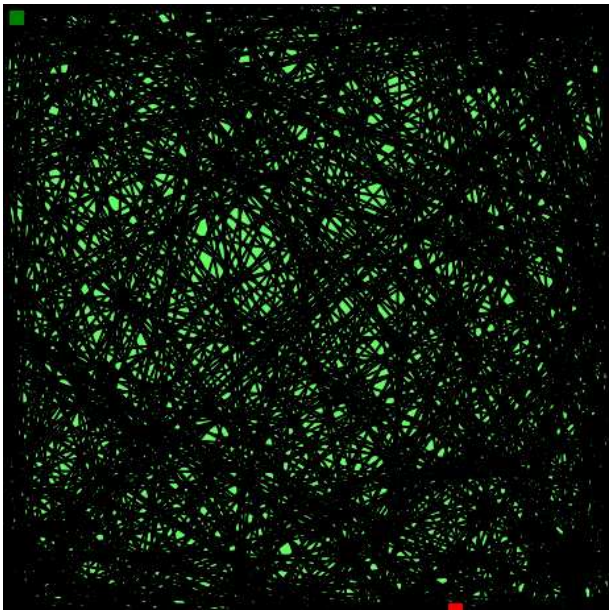
Vi vil her sammenligne effektiviteten af Random Direction og Distance Trasform Path Planning algoritmerne i forskellige typer områder, for at dokumentere, hvilken der er mest effektiv. DTPP er kun simuleret én gang, fordi den genererede rute vil blive den samme hver gang. Random Direction er derimod simuleret ti gange, da statistikken for denne kan svinge.

Forsøg 1

Hvis man tager udgangspunkt i det område, vores matlab simulering arbejdede med, tilbagelagde Random Direction gennemsnitlig et område, der var 14 gange så stort som det areal, der skulle dækkes. Ved at genskabe de samme forhold i vores program, og derefter simulere kørsel med DTPP og Random Direction, har vi beregnet, at DTPP kun tilbagelægger 1,167 gange det område, der skal dækkes, via en rute vist på figur 8.1, hvor Random Direction tilbagelægger 14,006 gange det pågældende område. I et kvadratisk område af denne størrelse er DTPP altså 12 gange mere effektiv end Random Direction. Dataene kan ses i tabel 8.1 og 8.2.



Figur 8.1 *DTPP's beregnede rute i et område lig det, der blev brugt i vores matlab test af RD (Se afsnit 2.5).*



Figur 8.2 *RD's beregnede rute i et område lig det, der blev brugt i vores matlab test af RD (Se afsnit 2.5).*

Koordinater for matlab simuleringområde		
(0;0) , (46;0) , (46;46) , (0;46)		
Parametre		
Startpunkt	(1;1)	
Stoppunkt	(2;1)	
Hastighed	1.00 m/s	
Bredde	0.8535 m	
Margin bredde	0.1465 m	
Vinkelhastighed	90 $^\circ/s$	
	DTPP	RD matlab
Rutelængde	2468.86 m	3,18252e5 pixel
Tid	43.78 min	-
Tilbagelagt areal	2468.86 m^2	3,500772e6 pixel
Områdeareal	2116.00 m^2	2,5e5 pixel
Effektivitet	0.857	0,071

Tabel 8.1 Parametre og statistik for simulerede ruter i det område der er afbilledet på figur 8.1.

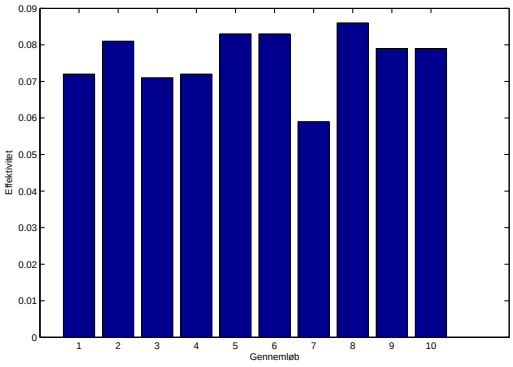
Forsøg 2

I et område med form som Aalborg Universitets gårdhaver, hvor de nuværende autonome plæneklippere anvendes, tilbagelægger DTPP 1,133 gange det areal, der skal dækkes, som set på figur 8.4, hvor Random Direction tilbagelægger 13,072 gange området. Her er DTPP altså 11,5 gange mere effektiv end Random Direction. Dataene kan ses i tabel 8.3 og 8.4.

Ligesom i det kvadratiske område, vi brugte til vores matlabtest, er der stor forskel mellem DTPP og RD ved denne type område. DTPP har en langt kortere rute og kører tydligvis mere effektivt. Den tilbagelægger blot 1,331 gange det areal der skal dækkes mod RDs 19,158 gange.

Koordinater for matlab simuleringområde					
(0;0) , (46;0) , (46;46) , (0;46)					
Parametre					
Startpunkt	(1;1)				
Stoppunkt	(2;1)				
Hastighed	1.00 m/s				
Bredde	1 m				
Vinkelhastighed	90 $^\circ/s$				
	Random Direction				
Simulering #	1	2	3	4	5
Rutelængde	29345.23m	26051.50m	29785.25m	2944.243m	25506.32m
Tid	517.76min	458.87min	523.66min	519.24min	449.44min
Tilbagelagt areal	29345.23m^2	26051.50m^2	29785.25m^2	26773.13m^2	26746.41m^2
Områdeareal	2116.00m^2	2116.00m^2	2116.00m^2	2116.00m^2	2116.00m^2
Effektivitet	0.072	0.081	0.071	0.072	0.083
Simulering #	6	7	8	9	10
Rutelængde	25576.19m	35866.01m	24576.03m	26773.13m	26746.41m
Tid	450.35min	630.59min	432.55min	470.61min	470.23 min
Tilbagelagt areal	25576.19m^2	35866.01m^2	24576.03m^2	26773.13m^2	26746.41m^2
Områdeareal	2116.00m^2	2116.00m^2	2116.00m^2	2116.00m^2	2116.00m^2
Effektivitet	0.083	0.059	0.086	0.079	0.079
Gennemsnilig effektivitet	0.0714				

Tabel 8.2 Parametre og statistik for 10 simulerede Random Direction-ruter i det område der er afbilledet på figur 8.2.

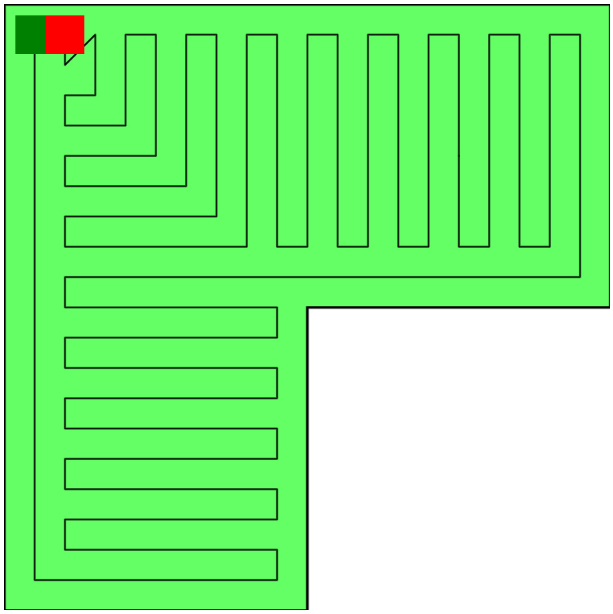


Figur 8.3 *Graf over effektiviteten af 10 gennemløb med Random Direction på et kvadratisk område.*

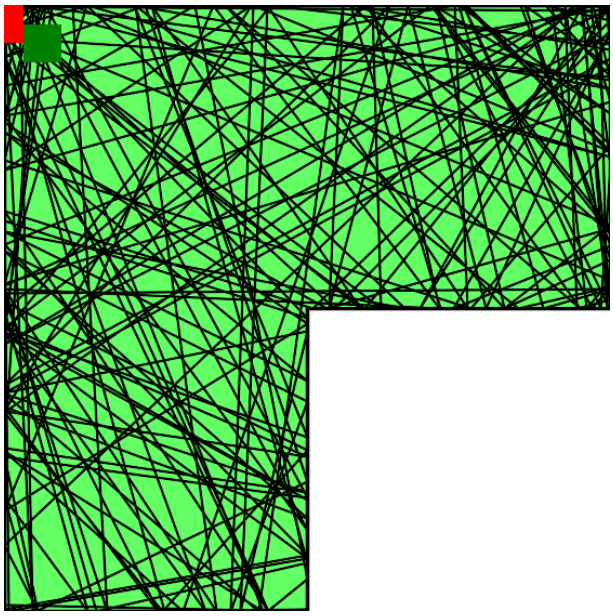
Forsøg 3

Begge algoritmer i vores program formår også at dække et mere komplekst område. Hvis man sammenligner statistik er det dog stadig DTPP, der kører mest effektivt. RD tilbagelægger 17,634 gange så stort et areal, som DTPP for at dække området. DTPP tilbagelægger 1,431 gange områdets alreal mod Random Directions 9,970 gange. Dataene kan ses på tabel 8.5 og 8.6.

Billede 8.7 viser DTPP's problematik i at lukke sig selv inde. Måden vi har løst problemet på er ved at undersøge om hele området er dækket. Hvis noget af området ikke er dækket vil den kører over til det nærmeste område som ikke er dækket, og derefter forsætte. Som det ses på billede 8.7



Figur 8.4 *DTPPs beregnede rute i et område som gårdhaverne på Aalborg Universitet.*



Figur 8.5 *RDs beregnede rute i et område som gårdhaverne på Aalborg Universitet.*

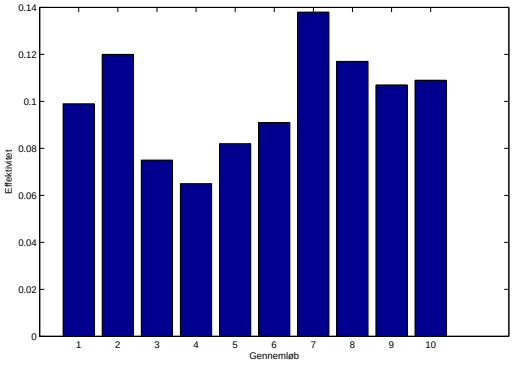
bevæger enheden sig uden for området, det er selvfølgelig ikke særlig hensigtsmæssig i den virkelige verden. Hvis DTPP skulle anvendes i en robot vil det kræve at udover at finde det nærmeste ikke dækkede areal, også vil være nødt til at undersøge i forhold til afstanden som den skal køre for ikke at bevæge sig uden for området.

Koordinater for gårdhave på Aalborg Universitet	
(0;0) , (16;0) , (16;8) , (8;8) , (8;16) , (0;16)	
Parametre	
Startpunkt	(1;1)
Stoppunkt	(2;1)
Hastighed	1.00 m/s
Bredde	0.8 m
Margin bredde	0.2 m
Vinkelhastighed	90 $^\circ/s$
	DTPP
Rutelængde	265.13 m
Tid	5.64 min
Tilbagelagt areal	265.13 m^2
Områdeareal	192.00 m^2
Effektivitet	0.724

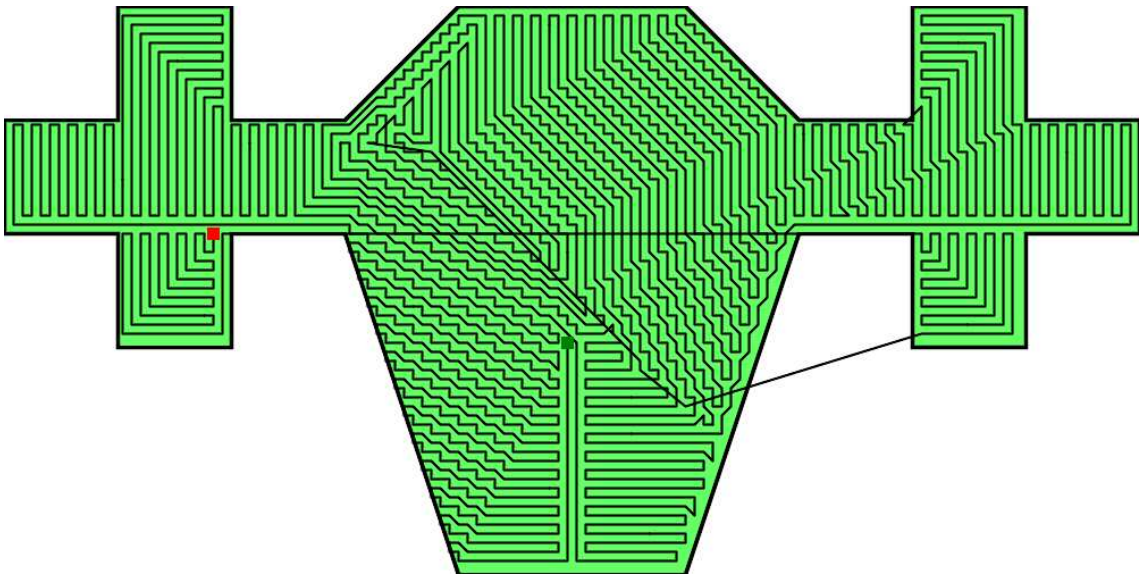
Tabel 8.3 Parametre og statistik for simulerede ruter i det område, der er afbilledet på figur 8.4

Koordinater for gårdhave på Aalborg Universitet					
(0;0) , (46;0) , (46;46) , (0;46)					
Parametre					
Startpunkt	(1;1)				
Stoppunkt	(2;1)				
Hastighed	1.00 m/s				
Bredde	1 m				
Vinkelhastighed	90 $^\circ/s$				
	Random Direction				
Simulering #	1	2	3	4	5
Rutelængde	1939.34m	1606.08m	2545.34m	2933.41m	2352.89m
Tid	39.16min	32.63 min	51.45min	58.87min	47.81min
Tilbagelagt areal	1939.34m^2	1606.08m^2	2545.34m^2	2933.41m^2	2352.89m^2
Områdeareal	192.00m^2	192.00m^2	192.00m^2	192.00m^2	192.00m^2
Effektivitet	0.099	0.120	0.075	0.065	0.082
Simulering #	6	7	8	9	10
Rutelængde	2107.43m	1393.54m	1646.01m	1786.36m	1764.45m
Tid	42.75min	28.09min	34.01min	36.40min	35.72min
Tilbagelagt areal	2107.43m^2	1393.54m^2	1646.01m^2	1786.36m^2	1764.45m^2
Områdeareal	192.00m^2	192.00m^2	192.00m^2	192.00m^2	192.00m^2
Effektivitet	0.091	0.138	0.117	0.107	0.109
Gennemsnilig effektivitet	0.0765				

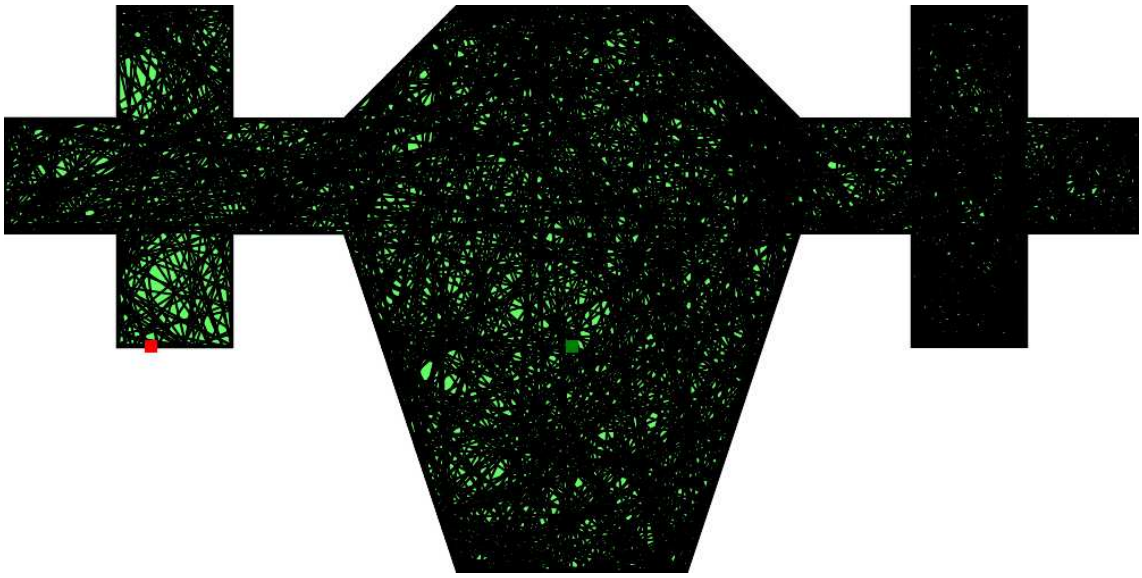
Tabel 8.4 Parametre og statistik for 10 simulerede Random Direction-ruter i det område der er afbilledet på figur 8.5.



Figur 8.6 *Graf over effektiviteten af 10 gennemløb med Random Direction på et område svarende til gårdhaven på Aalborg Universitet.*



Figur 8.7 *Den beregnede rute i et komplekst område med DTPP.*



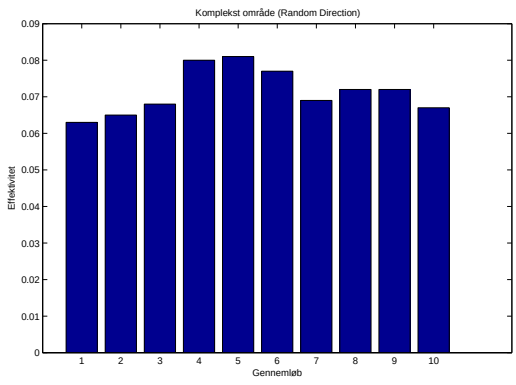
Figur 8.8 *Den beregnede rute i et komplekst område med RD.*

Koordinater for komplekst område	
(20;0) , (20;20) , (0;20) , (0;40) , (20;40) , (20;60) , (40;60) , (40;40) , (60;40) , (80;100) , (120;100) ,(140;40) , (160;40) , (160;60) , (180;60) , (180;40) , (200;40) ,(200;20) , (180;20) , (180;0) , (160;0) , (160;20) , (140;20) ,(120;0) , (80;0) , (60;20) , (40;20) , (40;0)	
Parametre	
Startpunkt	(100;60)
Stoppunkt	(70;20)
Hastighed	1.00 m/s
Bredde	1.7 m
Margin bredde	0.3 m
Vinkelhastighed	90 $^\circ/s$
	DTPP
Rutelængde	7438.33m
Tid	163.89 min
Tilbagelagt areal	14876.66 m^2
Områdeareal	10400.00 m^2
Effektivitet	0.699

Tabel 8.5 Parametre og statistik for simulerede ruter i det område der er afbilledet på figur 8.7.

Koordinater for komplekst område					
(0;0) , (46;0) , (46;46) , (0;46)					
Parametre					
Startpunkt	(1;1)				
Stoppunkt	(2;1)				
Hastighed	1.00 m/s				
Bredde	2 m				
Vinkelhastighed	90 $^\circ/s$				
	Random Direction				
Simulering #	1	2	3	4	5
Rutelængde	78121.08m	76108.24m	73263.37m	62014.27m	61126.68m
Tid	1364.86min	1331.44min	1278.79min	1077.98min	1065.10min
Tilbagelagt areal	164054.27m^2	159827.29m^2	153853.07m^2	130229.97m^2	128366.02m^2
Områdeareal	10400.00m^2	10400.00m^2	10400.00m^2	10400.00m^2	10400.00m^2
Effektivitet	0.063	0.065	0.068	0.080	0.081
Simulering #	6	7	8	9	10
Rutelængde	64251.56m	71486.77m	69067.61m	68357.08m	74179.41m
Tid	1121.42min	1248.80min	1206.88min	1191.75min	1294.96min
Tilbagelagt areal	134928.27m^2	150122.20m^2	145041.98m^2	143549.86m^2	155776.74m^2
Områdeareal	10400.00m^2	10400.00m^2	10400.00m^2	10400.00m^2	10400.00m^2
Effektivitet	0.077	0.069	0.072	0.072	0.067
Gennemsnilig effektivitet	0.1003				

Tabel 8.6 Parametre og statistik for 10 simulerede Random Direction-ruter i det område der er afbilledet på figur 8.8.



Figur 8.9 *Graf over effektiviteten af 10 gennemløb med Random Direction på et komplekst område.*

Kapitel 9

Konklusion

I vores problemformulering har vi stillet følgende spørgsmål:

- Hvilken rutealgoritme er mest effektiv: Random Direction eller Distance Transform Path Planning?
- I hvilke situationer er Random Direction bedre end Distance Transform Path Planning?
- I hvilke situationer er det modsatte tilfældet?

I kapitel 8 har vi vist, at Random Direction er ca. $\frac{1}{12}$ så effektiv som Distance Transform Path Planning. Det vil sige, at Random direction skal køre en rute, der er ca. 12 gange længere end den teoretisk nødvendige, for at dække arealet. Hvorimod DTPP kører en rute, der er meget tæt på den kortest mulige rute, og derfor er DTPP den mest effektive algoritme af de to.

Der er en vis risiko for, at en autonom plæneklipper med en ruteberegning ud fra DTPP algoritmen, vil lukke sig selv inde i et hjørne el.lign., og dermed stoppe inden hele arealet er dækket. Dette har vi dog taget højde for i vores program, så den eneste ulempe ved dette vil være, at den krydser enkelte områder flere gange, når den skal bevæge sig fra en blindgyde, til de områder, som endnu ikke er blevet dækket. Random Direction algoritmen vil aldrig have problemer med, at den lukker sig inde, men tilgengæld kører den samme sted op til flere gange.

Den endelige konklusion må derfor være, at DTPP i høj grad vil kunne effektivisere autonome plæneklipperes arbejdsgang, og derfor er den, den mest effektive rutealgoritme af de to vi har undersøgt i vores projekt.

9.1 Perspektivering

Vi har valgt, at udforme vores perspektivering som en teknologivurdering, hvor vi bruger SWOT-analyse metoden/footnoteVi har fået kenskab til SWOT metoden via vores vejledere, og beskrivelser som http://www.startogvaekst.dk/sw431.asp. SWOT-analysen bliver brugt ude i erhvervslivet, til at få et overblik over en virksomheds status på et givent tidspunkt. Derefter kan virksomheden bruge resultaterne fra SWOT-analysen, til at lave en strategi for fremtiden. I vores tilfælde laver vi en analyse på vores produkt.
En SWOT -analyse er delt op i fire områder: Strenghts, Weaknesses, Opportunities, og Threats. [1].
De fire områder kan beskrives som følger:

- **Styrker** Parametre der kan hjælpe en, til at opnå et mål eller en ambition.
- **Svagheder** Parametre der kan forhindre en, i at opnå et mål eller en ambition.
- **Muligheder** Interne parametre der kan hjælpe en, til at opnå et mål eller en ambition.
- **Trusler** Eksterne parametre der kan forhindre en, i at opnå et mål eller en ambition.

De fire områder deles derefter op i par, så de viser følgende:
Virksomhedens indre formåen (Styrke- Svagheder)
Omgivelsernes indvirkning på virksomheden (Muligheder-Trusler)

9.1.1 SWOT-analyse på vores produkt

- **Strenghts**
  DTPP forkorter væsentligt den tid, som en autonom plæneklipper skal bruge på, at slå græs i. plæneklipperen behøver kun køre over det samme sted én gang, for at dække hele arealet, til forskel fra random direction, som kører samme sted flere gange, før at hele arealet er dækket. Simulerings programmet kan også udvides til, at kunne bruge flere rutealgoritmer, så man nemt vil kunne sammenligne forskellige algoritmers effektivitet.
- **Weaknesses**
  Vi har ikke en ny firmware til autonome plæneklippere klar endnu. Der er meget arbejde tilbage, hvis man skal implementere DTPP i en autonom plæneklipper. Vores simuleringsprogram har også den svaghed, at det ikke understøtter ”forhindringer”, som man kan forvente at støde på, på en rigtig græsplæne, som for eksempel træer, blomsterbede, havestole, og så videre.
- **Opportunities**
  De autonome plæneklipperess nuværende ruteplanlægninssystem er en af deres største svagheder, og vil blive forbedret på et tidspunkt. Derfor vil det være en fordel for en producent at være på forkant med udviklingen, ved at investere i et navigationssystem og ruteplanlægningssystem før konkurrenterne. Vi kan se, at det vil tage kortere tid at slå græsset, og at det vil kunne give forbrugeren nogle besparelser i tid og driftsomkostninger. Vores rutesimulering er ikke kun noget, der kan bruges til autonome plæneklippere. Den vil også gælde for andre autonome køretøjer. For eksempel golfbold-opsamler, minerydderer, og så videre. Stort set alle slags maskiner, som i deres arbejde skal dække et givet areal. Hvis vi havde mere tid til dette projekt, ville vi kunne lave det stykke firmware som vil kunne instaleres i en autonom plæneklipper.
- **Threats**
  Autonome plæneklippere er forholdsvis dyre i forhold til de konventionelle plæneklippere, som forbrugerne er vant til, og ikke alle mennesker er parat, til at skulle have en robot i haven. Markedet ikke er særligt stort, og det er derfor ikke sikkert, at virksomhederne vil investere i forbedrede ruteplanlægningssystemer. Endvidere er der risiko for, at et andet firma

[1] Styrker, Svagheder, Muligheder, Trusler

vil udvikle et bedre/billigere system end os. Et ruteplanlægningssystem, som DTPP, kræver et navigationssystem. De autonome plæneklippere som findes i dag, har ikke den nødvendige hardware, til at kunne navigere præcist. En implementation af en forbedret rutealgoritme vil derfor udgøre en ekstra udgift til produktionen, ud over de ressourcer som i forvejen skal bruges til selve implementationen.

Appendiks A

Definitioner

Ord	Definition
Autonom plæneklipper	Selvkørende plæneklipper, som for eksempel Robomow og Auto Mower, der kan slå græs uden menneskelig indblanding.
Effektivitet	Arealet af det område som skal klippes, divideret med det areal som plæneklipperen har tilbagelagt.
Tilbagelagt areal	Defineres som rutelængde · klippebredde

Appendiks **B**

Ordliste

Ord	Betydning
Galileo	Positionerings system
GPS	Global Positioning System
SWOT	Strengths, Weaknesses, Opportunities og Threats; Oversat: Styrker, Svagheder, Muligheder og Trusler
SPU	Struktureret Program Udvikling

Appendiks C

Interviewguide

Overvejelser om anskaffelse

- Hvordan blev arbejdet udført før anskaffelsen?
- Hvilke overvejelser har I gjort jer inden anskaffelsen?

Fordele og ulemper ved autonom plæneklipper

- Er der problemer med uklippede pletter?
- Skal græsset trimmes af en person efterfølgende?
- Fordele?
- Ulemper?

Vedligehold og betjening

- Hvor stort et areal skal maskinerne dække?
- Hvor lang tid er maskinerne, om at dække dette areal?
- Hvilket mærke og hvilken model benytter I?
- Hvor mange autonome plæneklippere har I?
- Forslag til forbedringer?

Appendiks D

Matlab-script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title:       %  Random coverage efficincy calculater                    %
%              %                                                          %
% Author(s):   %  A302 Jesper Pedersen, Christoffer Poulsen, Søren Hede, %
%              %  Søren Thorup, Jesper Skjødt og Claus Pedersen          %
%              %                                                          %
% Date:        %  18/04/2006                                              %
%              %                                                          %
% Discription: %  A matlab program for calculating the effenciency of    %
%              %  the coverage of a square area, when moving in a        %
%              %  random direction every time the baoundry is hit.       %
%              %  Lines are drawn by creating random points around       %
%              %  the boundry.                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear
clc
% An "empty" matrix is created
for m = 1:500
for n = 1:500
empty(m,n) = 120;
end;
end;
empty = uint8(empty);
%imwrite(empty,'billede.gif')
img = empty;
tempimg = empty;
stregareal(1) = 0;


% Here the first point is determined
if rand(1)*10 < 5
if rand(1)*10 < 5
x(1) = 0
y(1) = rand(1)*500
else
x(1) = 500
y(1) = rand(1)*500
end;
elseif rand(1)*10 < 5
y(1) = 0
```

```
x(1) = rand(1)*500
else
y(1) = 500
x(1) = rand(1)*500
end;

%Here the rest of the points are determined. The script makes sure no two following
points lie on the same boundry and choses randomly between the three other sides.
%for k =2:100
k = 2;
kk = 1;
while stregareal(kk) ~= 500 * 500
if x(k-1) == 0
randomnr = rand(1)*10
if randomnr < 10/3
x(k) = 500;
y(k) = rand(1)*500;
elseif randomnr < 20/3
y(k) = 500;
x(k) = rand(1)*500;
else
y(k) = 0;
x(k) = rand(1)*500;
end
elseif x(k-1) == 500
randomnr = rand(1)*10
if randomnr < 10/3
x(k) = 0;
y(k) = rand(1)*500;
elseif randomnr < 20/3
y(k) = 500;
x(k) = rand(1)*500;
else
y(k) = 0;
x(k) = rand(1)*500;
end
elseif y(k-1) == 500
randomnr = rand(1)*10;
if randomnr < 10/3
y(k) = 0;
x(k) = rand(1)*500;
elseif randomnr < 20/3
x(k) = 500;
y(k) = rand(1)*500;
else
x(k) = 0;
y(k) = rand(1)*500;
end
elseif y(k-1) == 0
randomnr = rand(1)*10;
if randomnr < 10/3
y(k) = 500;
x(k) = rand(1)*500;
elseif randomnr < 20/3
x(k) = 500;
y(k) = rand(1)*500;
```

```
else
x(k) = 0;
y(k) = rand(1)*500;
end
end
k = k +1;
%end


%The lines are drawn.
%for kk = 1:99
tempimg = empty;

moo = x(kk+1)-x(kk);
if moo == 0
for jj = 1:499
img(jj,kk)=0;
end
else
a = (y(kk+1)-y(kk))/moo;
b = y(kk) - a * x(kk);
alpha(kk)=a;
beta(kk)=b;

%The tol-variable makes sure the lines are equally wide.

tol(kk)=5/(sin(0.5*pi-atan(a)));
for m = 0:499
for n = 0:499
F = a * (n) + b;
if m - tol(kk) <= F && F <= m + tol(kk)
img(m+1,n+1)=0;
tempimg(m+1,n+1)=0;
end;
end;
end;
end;

stregareal(kk+1) = length(find(img == 0));
tempstreger(kk) = length(find(tempimg == 0));
if kk ~= 1
streger(kk) = stregareal(kk + 1) - stregareal(kk);
else
streger(kk) = stregareal(kk + 1);
end;
eff(kk) = streger(kk) / tempstreger(kk) * 100;
kk = kk +1
%imwrite(img,'billede.gif','WriteMode','append')
end;

%the length of the lines are calculated
length=0;
for hh = 1:kk-1
length = length + sqrt((y(hh+1)-y(hh))^2 + (x(hh+1)-x(hh))^2);
end;
length
```

```
%omraade = length(find(img == 120))
%streger = length(find(img == 0))
%alpha
%beta
image(img)
t = 0:kk-2;
filtereff = lavpasfilter(eff,1,10,100);
plot(t,eff,t,filtereff,'linewidth',2)
xlabel('linje nr.')
ylabel('%')
```

Appendiks E

Kildekritik

Vi har valgt at lave kildekritik på www.wikipedia.org og GPSnet.dk, som begge optræder som kilder i vores rapport. Vi har valgt disse, da vi mener at det er de kilder der nemmest kan sættes spørgsmålstegn ved.

Wikipedia
http://en.wikipedia.org/

Wikipedia er et online opensource leksikon, hvor brugerne selv tilføjer, opdaterer, og retter artiklerne. Derfor har artiklerne ofte adskillelige anonyme forfattere, hvilket gør at troværdigheden og faktualiteten af indholdet kan fremstå tvivlsom. Der har siden wikipedias start i 2001 også været enkelte sager, hvor artikler med politisk og religiøst indhold, har måtte "låses", for at forhindre brugere, i at kunne ændre artiklerne, til at fremhæve deres personlige meninger og overbevisninger.
Dog viser en undersøgelse fra det engelske magazin Nature[1], at wikipedias naturvidenskabelige artikler har næsten lige så få fejl som Encyclopedia Britannica, som regnes for et af de bedste leksikon i verden. I artiklen står der:
" ...Wikipedia comes close to Britannica in terms of the accuracy of its science entries"

Da vi udelukkende har brugt naturvidenskablige artikler fra wikipedia, og fordi vi ikke har brugt wikipedia som nogen stor kilde, konkluderer vi derfor, at de usikkerheder der måtte være, ikke har haft nogen indflydelse på vores projekt.

GPSnet.dk
http://www.gpsnet.dk/

GPSnet.dk, er en dansk hjemmeside, som tilbyder mere nøjagtig possitions-bestemmelse til folk der kræver mere præcition end almindelig GPS. Da GPSnet.dk også fremstår som sælger af dette produkt, kunne man fristes til at tro, at de nemt kan have pyntet på deres tal, over hvor præcis man via dem kan bestemme geografiske possitioner. Men da den teknologi der benyttes er brugt andre steder i verden, og de har brugere af systemet, som for eksempel Vejdirektoratet og DTU, vuderer vi at de informationer vi har fået fra deres hjemmeside er troværdige.

[1] Uddrag af artiklen kan findes på http://www.nature.com/news/2005/051212/full/438900a.html

Litteratur

[1] Computer Networks
Tanenbaum, Andrew S.
2003 Pearson Education Inc.
Upper Saddle River New Jersey 07458

[2] (Robomow_Med_ladestation.pdf)
side 35, 2.6 Indstiling af land og kalibrering af kompas
side 41, 3.8 Scanning (klipning)

[3] (DK_OM_2006.pdf)
side 15, 2. Præsentation - Bevægelsesmønster

[4] Dept. of math and computer sciences
A survey of mobility models for ad hoc network research
Tracy Camp, Jeff Boleng, Vanessa Davies

[5] Struktureret Programudvikling
Stephen Biering-Sørensen, Finn Overgaard-Hansen, Susanne Klim, Preben Thalund Madsen
Nyt Teknisk Forlag, ISBN 87-571-1046-8

[6] (pos96rep.pdf)
Where am I? Sensors and methods for mobile Robot positioning
ftp://www.eecs.umich.edu/people/johannb/pos96rep.pdf
J. Borenstein, H. R. Everett, and L.Feng
Edited and compiled by J. Borenstein, April 1996

[7] (Global Positioning System - Wikipedia.pdf)
Global Positioning System
http://en.wikipedia.org/wiki/Gps
From Wikipedia, the free encyclopedia

[8] (GPS-X-02007.pdf)
GPS Basics
http://www.u-blox.com/technology/GPS-X-02007.pdf
Jean-Marie Zogg, 2002

[9] (Galileo positioning system - Wikipedia.pdf)
Galileo positioning system
http://en.wikipedia.org/wiki/Galileo_positioning_system
From Wikipedia, the free encyclopedia

[10] (Navigation - Farvandsvæsenet.pdf)
Farvandsvæsenet
http://www.fomfrv.dk/om_frv/hovedopgaver/navigation.htm

[11] (DGPS service fra GPSnet.dk.pdf)
DGPS service fra GPSnet.dk
http://gpsnet.dk/showpage.php?nID=123

[12] (GPSnet.dk.pdf)
VRS service fra GPSnet.dk
http://www.gpsnet.dk/showimg.php?ID=601

Internetkilder nævnt i literaturlisten er vedlagt som pdf på bilag F.1. Filnavnene er angivet i parantes.

Appendiks F

Bilag

F.1 CD

Indhold

Kilder i pdf format
DGPS service fra GPSnet.dk.pdf
DK_OM_2006.pdf
Galileo positioning system - Wikipedia.pdf
Global Positioning System - Wikipedia.pdf
GPS-X-02007.pdf
GPSnet.dk.pdf
Navigation - Farvandsvæsenet.pdf
pos96rep.pdf
Robomow_Med_ladestation.pdf

Matlab-script i zip-fil
Matlab.zip

Folder med kildekode til vores program
..\ program :
brugerinput.c
brugerinterface.c
DTPP-ml-area
dtpp.c
filhaandtering.c
list.c
omraade.c
partyelg
rd.c
rutesimulering.c
simulering.c
statistik.c
testdriver-simulering.c
uni-have-area
visualisering.c