

CDIO-projekt

LEGO-robot

Gruppe nr.: 4
Rapport nr.: Endelig rapport
Livscyklus fase: Færdig
Plan status: Følges



Afleveret via CampusNet

s042067 - Clausen, Per Boye
Initialer: PC — Projektleder



Afleveret via CampusNet

s070162 - Kronborg, Jeppe
Initialer: JK



Afleveret via CampusNet

s093482 - Brix, Terkel Thorbjørn
Initialer: TB



Afleveret via CampusNet

s083117 - Andersen, Morten Hulvej
Initialer: MA — Stedfortræder



Afleveret via CampusNet

s093478 - Hansen, Mathias
Initialer: MH

Indholdsfortegnelse

1 Indledning	1
1.1 Problemformulering	1
1.2 Konkurerencen	1
2 Analyse	2
2.1 Krav	2
2.2 Successkriterier	2
2.3 Mål	2
2.4 Lösungsstrategi/Projektplan	2
2.4.1 Robot	2
2.4.2 Billedbehandling	2
2.4.3 Stifinding	2
3 Robot	3
3.1 Styring	3
3.2 Kommunikation/RMI	3
3.3 Process/Iteration	3
3.4 Test	3
3.5 Videreudvikling	3
4 Billedbehandling	4
4.1 Design	4
4.1.1 Webcam	4
4.1.2 Billedbehandling	5
4.2 Implementering	6
4.2.1 Webcam	6
4.2.2 Billedbehandling	6
4.3 Test	7
4.4 Udviklingsproces	7

5 Implementering	8
5.1 Valg af algoritme	8
5.2 Implementering	8
5.3 Optimering	9
6 Samling	10
6.1 Robot	10
6.2 Billedbehandling	10
6.2.1 Webcam	10
6.3 Stifinding	10
6.4 Videreudvikling	10
7 Process	11
7.1 Ansvarsfordeling.	11
7.2 Kvalitet	11
8 Konklusion	12
8.1 Konkurrencen	12
Bilag.	A-1

Figurer

Kapitel 1

Indledning

1.1 Problemformulering. 1

1.2 Konkurerencen 1

Problemformulering 1.1

Konkurerencen 1.2

Kapitel 2

Analyse

2.1 Krav	2
2.2 Successkriterier	2
2.3 Mål	2
2.4 Løsningsstrategi/Projektplan	2
2.4.1 Robot	2
2.4.2 Billedbehandling	2
2.4.3 Stifinding	2

Krav	2.1
------	-----

Successkriterier	2.2
------------------	-----

Mål	2.3
-----	-----

Løsningsstrategi/Projektplan	2.4
------------------------------	-----

2.4.1 Robot

2.4.2 Billedbehandling

2.4.3 Stifinding

Kapitel 3

Robot

3.1 Styring	3
3.2 Kommunikation/RMI	3
3.3 Process/Iteration	3
3.4 Test	3
3.5 Videreudvikling	3

Styring	3.1
---------	-----

Kommunikation/RMI	3.2
-------------------	-----

Process/Iteration	3.3
-------------------	-----

Test	3.4
------	-----

Videreudvikling	3.5
-----------------	-----

Kapitel 4

Billedbehandling

4.1 Design	4
4.1.1 Webcam	4
4.1.2 Billedbehandling	5
4.2 Implementering	6
4.2.1 Webcam	6
4.2.2 Billedbehandling	6
4.3 Test	7
4.4 Udviklingsproces	7

Design

4.1

Billedbehandlingen inddeles i to dele: Webcam og behandling af billedet.

4.1.1 Webcam

Webcam-delen sørger for at varetage forbindelsen til webcam, og herfra hentes rå billeder.

Der defineres et interface, `IImageSource`, som specificerer metoderne `init()`, som benyttes til at forbinde til webcam – `close()`, som lukker forbindelsen til webcam – samt `getImage()`, som returnerer et billede fra webcam som et `BufferedImage` objekt.

Til at håndtere selve forbindelsen til kameraet benyttes JMF **cite**.

4.1.2 Billedbehandling

Billedbehandlingen behandler billedet fra kameraet og bestemmer positioner af forhindringer, kager og robotter, nærmere bestemt:

Fortolkning af kildebillede til et *tilemap*¹. Hver pixel undersøges i forhold til fastsatte grænseværdier.

Filtrering af *tilemap*, hvor områder af pixels sorteres fra, hvis ikke de dækker et tilstrækkelig stort antal sammenhængende pixels. Dette fjerner støj fra billedet, og sikrer mod fejlagtig genkendelse af objekter.

Bestemmelse af grænser for banen ud fra de 4 hjørne-forhindringer.

Generering af map af forhindringer, hvor der er tilføjet buffer-zoner omkring forhindringer. Hvis det ønskes, kan robot 2 her markeres, ligeledes med en buffer-zone, i tilfælde, hvor robot 1 skal finde vej uden om.

Bestemmelse af position for kager.

Bestemmelse af position og vinkel for robotter.

Skalering af output for at optimere køretiden for stifindingen. Denne funktionalitet er ikke taget i brug.

Generering af grafisk repræsentation af de behandlede data, så det er muligt at følge billedbehandlingsarbejdet løbende.

Opbygning

Der er specificeret et interface – `IImageProcessor` – som billedbehandlingen skal implementere. I dette interface lægges også standard-værdier for mange af de parametre, som billedbehandlingen gør brug af.

Selve billedbehandlingen er implementeret i `ImageProcessor2` klassen. **Forklaring på 2?** Al funktionalitet er specificeret her.

Data, som skal benyttes videre i det samlede system, returneres i `DTO`² klassen `Locations`, som implementerer `ILocations` interfacet. Disse entiteter er rent databærende.

`ILocations` gemmer *tilemap* og forhindrings *map* som 2D int-arrays, og kager og robotter gemmes som lister af hhv. `Cake`- og `Robot` `DTO`-objekter. Kildebillede og fortolket billede gemmes som `BufferedImage` objekter.

¹Tilemap er et 2D-array af integers, som er opbygget som et billede med meget få farver (én farve pr. genkendt objekttype)

²Data Transfer Object

Implementering

4.2

4.2.1 Webcam

Der benyttes fortrinsvis JMF til hele implementeringen af webcam forbindelsen. Implementeringen tager udgangspunkt i det eksempel, som er givet på CampusNet.

Ref?

I `init()` metoden oprettes der forbindelse til enheden „vfw:Microsoft WDM Image Capture (Win32):0“. Der benyttes formatet 320x240 RGB.

Selve forbindelsen bruges gennem et statisk `Player` objekt.

4.2.2 Billedbehandling

I billedbehandlingen ligger klassevariable med tilhørende get-/set-metoder til de parametre, som er specificeret i `IIImageProcessor`.

examineImage(...) Denne metode benyttes, når et billede skal behandles. Metoden tager som argumenter det kildebillede (et `BufferedImage`), som skal behandles, samt en boolsk værdi, som dikterer, hvorvidt en grafisk repræsentation af det behandlede billede skal dannes.

Metoden returnerer et `Locations` objekt.

`examineImage(...)` benytter de øvrige metoder i billedbehandlingen til at behandle det givne kildebillede.

generateTilemap() Her dannes ud fra kildebilledet et 2D-array med samme størrelse. Hver pixel undersøges i forhold til **Threshold** objekter. Der tjekkes i rækkefølgen *forhindring*, *kage*, *robot 1 (front-bag)*, *robot2 (front-bag)*. Hvis ikke en pixel bliver genkendt her, tolkes den som værende baggrund/gulv.

Metoden gemmer det resulterende *tilemap* i klasse-variablen `tilemap`.

filterObstacles() Filtrering af forhindrings-pixels foregår her. Samtidig foretages første generering af forhindrings map – endnu et 2D-array, `obstaclemap`, med samme størrelse som `tilemap`.

Der benyttes endnu et 2D-array til at holde styr på behandlede pixels.

Metoden vandrer igennem alle pixels i `tilemap`. Hver gang en forhindring, som ikke allerede er behandlet, registreres, benyttes hjælpemetoden **collectRecursion()** til at samle alle de sammenhængende forhindrings-pixels.

Hvis antallet af opsamlede koordinater er mindre end grænsen `MIN_OBJECT_SIZE`, forkastes forhindringen, og de fundne pixels defineres som baggrund i `tilemap` og

`obstaclemap`.

I modsat fald – hvis den opsamlede forhindring er tilstrækkelig stor – registreres de opsamlede punkter i `obstaclemap`.

findBounds() Grænserne for selve banen bestemmes groft ved at finde den øverste og nederste vandrette linje i `tilemap`, hvor der er min. 5 forhindrings-pixels. Tilsvarende gælder for lodrette linjer mod venstre og højre.

De fundne grænseværdier gemmes som `int`-array på formen { *top, venstre, bund, højre* }.

Test 4.3

Udviklingsproces 4.4

Kapitel 5

Implementering

5.1 Valg af algoritme	8
5.2 Implementering.	8
5.3 Optimering.	9

Valg af algoritme 5.1

Efter nøje research af mulige algoritmer er valget faldet på **A*** algoritmen. Algoritmen er valgt på baggrund af tidligere erfaringer fra nogle af gruppens medlemmer samt fordi at den opfylder projektets behov i forhold til effektivitet og brugbarhed. Desuden er algoritmen meget udbredt og har vist sit værd i utallige software projekter tidligere.

Dijkstra's algoritme har også været overvejet, da denne algoritme kunne bruges som base for en videreudvikling i forhold til vores krav, dette blev dog droppet da det ikke regnedes for, ikke at kunne betale sig.

Implementering 5.2

Stifindingen blev implementeret som en separat pakke i projektet med dertilhørende klasser der henholdsvis repræsenterer `TileMap`, `Path` og `Steps`. Implementeringen er baseret på en eksisterende implementeringen af *Kevin Glass*.

Dertil er implementeringen blevet optimeret og justeret til projektets behov.

Optimering

5.3

Algoritmen er optimeret således at antallet af steps der returneres af `findPath(...)` metoden er reduceret til et minimum, dette er gjort for at hjælpe controlleren i arbejdet med at instruere robotens bevægelser. Antallet af steps er reduceret ved at loope igennem alle steps, og for hvert step benytte *afstandsformlen* og kun tage step'et med i path'en såfremt afstanden til det forrige valgte step er over en foruddefineret grænseværdi.

Kapitel 6

Samling

6.1 Robot	10
6.2 Billedbehandling	10
6.2.1 Webcam	10
6.3 Stifinding	10
6.4 Videreudvikling	10

Robot	6.1
-------	-----

Billedbehandling	6.2
------------------	-----

6.2.1 Webcam

Stifinding	6.3
------------	-----

Videreudvikling	6.4
-----------------	-----

Kapitel 7

Process

7.1 Ansvarsfordeling 11

7.2 Kvalitet 11

Ansvarsfordeling 7.1

Kvalitet 7.2

Kapitel 8

Konklusion

8.1 Konkurrencen	12
Bilag	A-1
A.1 Gruppens medlemmer	A-2
A.2 Kommunikation	A-2
A.3 Møder	A-3
A.4 Roller	A-3
A.5 Dispositioner	A-3
A.6 Dokumenthåndtering	A-3
B.1 Tidsplan	A-4
B.2 Risici	A-4
B.3 Projektplan	A-4
C.1 Gruppens medlemmer	A-5
C.2 Kommunikation	A-5
C.3 Møder	A-6
C.4 Roller	A-6
C.5 Dispositioner	A-6
C.6 Dokumenthåndtering	A-6

Konkurrencen

8.1

Bilag

Indholdsfortegnelse

A Gruppekontrakt	A-2
A.1Gruppens medlemmer	A-2
A.2Kommunikation	A-2
A.3Møder	A-3
A.4Roller	A-3
A.5Dispositioner	A-3
A.6Dokumenthåndtering.	A-3
B Status Rapport Final.	A-4
B.1Tidsplan	A-4
B.2Risici	A-4
B.3Projektplan	A-4
C Gruppekontrakt	A-5
C.1Gruppens medlemmer	A-5
C.2Kommunikation	A-5
C.3Møder	A-6
C.4Roller	A-6
C.5Dispositioner	A-6
C.6Dokumenthåndtering.	A-6

Bilag A

Gruppekonsrakt

Gruppens medlemmer

A.1

JK Jeppe Kronborg, s070162

PC Per Boye Clausen, s042067

TB Terkel Brix, s093482

MA Morten Hulvej Andersen, s083117

MH Mathias Hansen, s093478

Kommunikation

A.2

Korte/vigtige beskeder sendes gennem CampusNet gruppe som høj prioritet. Denne gruppe skal hos alle være indstillet til at sende høj-prioritet beskeder som SMS.

Løbende udvikling rapporteres gennem versionsstyringens push-notices.

Generelt udveksles løbende information gennem mødes; hver planlagt fælles-aktivitet begyndes med startmøde, og afsluttes med gå-hjem møde.

Afbud ifm. fælles aktivitet meldes hurtigst muligt som højprioritet besked på Campus-Net – alternativt direkte til projektleder – senest ved mødestart. For sent afbud noteres som fravær med note.

Møder

A.3

I 13-ugers perioden er mødetiden som udgangspunkt hver onsdag kl. 8.15-12, med startmøde kl. 10.

Øvrige arbejdstider aftales løbende, og hvert onsdags-startmøde tager stilling til individuel indsats.

Ved hvert møde udarbejdes en mødelog ud fra skabelon.

Roller

A.4

Projektleder:	PC
Stedfortræder:	MA
Materialeansvarlig:	JK
Dokument-ansvarlig:	MA

Dispositioner

A.5

Rapporter udarbejdes i \LaTeX . Til programmering er overordnet valgt Java.

Dokumenthåndtering

A.6

Mødelogs:	Dropbox
Kildekode:	git
Rapport:	git
Diverse dokumenter:	Dropbox

Bilag B

Status Rapport Final

Tidsplan	B.1
-----------------	------------

Risici	B.2
---------------	------------

Projektplan	B.3
--------------------	------------

Bilag C

Gruppekonsrakt

Gruppens medlemmer

C.1

JK Jeppe Kronborg, s070162

PC Per Boye Clausen, s042067

TB Terkel Brix, s093482

MA Morten Hulvej Andersen, s083117

MH Mathias Hansen, s093478

Kommunikation

C.2

Korte/vigtige beskeder sendes gennem CampusNet gruppe som høj prioritet. Denne gruppe skal hos alle være indstillet til at sende høj-prioritet beskeder som SMS.

Løbende udvikling rapporteres gennem versionsstyringens push-notices.

Generelt udveksles løbende information gennem mødes; hver planlagt fælles-aktivitet begyndes med startmøde, og afsluttes med gå-hjem møde.

Afbud ifm. fælles aktivitet meldes hurtigst muligt som højprioritet besked på Campus-Net – alternativt direkte til projektleder – senest ved mødestart. For sent afbud noteres som fravær med note.

Møder

C.3

I 13-ugers perioden er mødetiden som udgangspunkt hver onsdag kl. 8.15-12, med startmøde kl. 10.

Øvrige arbejdstider aftales løbende, og hvert onsdags-startmøde tager stilling til individuel indsats.

Ved hvert møde udarbejdes en mødelog ud fra skabelon.

Roller

C.4

Projektleder:	PC
Stedfortræder:	MA
Materialeansvarlig:	JK
Dokument-ansvarlig:	MA

Dispositioner

C.5

Rapporter udarbejdes i \LaTeX . Til programmering er overordnet valgt Java.

Dokumenthåndtering

C.6

Mødelogs:	Dropbox
Kildekode:	git
Rapport:	git
Diverse dokumenter:	Dropbox