Danmarks Tekniske Universitet

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

s083117 - Andersen, Morten Hulvej
Initialer: MA — Stedfortræder

Afleveret via CampusNet

s070162 - Kronborg, Jeppe
Initialer: JK

Afleveret via CampusNet

s093478 - Hansen, Mathias
Initialer: MH

Afleveret via CampusNet

s093482 - Brix, Terkel Thorbjørn
Initialer: TB

DTU Informatik
Institut for Informatik og Matematisk Modellering

Indholdsfortegnelse

Figurer

Kapitel 1

Billedbehandling

Design 1.1

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

1.1.1 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 fobinddelsen til kameraet benyttes JMF **cite** .

1.1.2 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 billedbehandlingens arbejde 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 billedet gemmes som `BufferedImage` objekter.

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

Implementering 1.2

1.2.1 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.

1.2.2 Billedbehandling

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

`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 1.3

Udviklingsproces 1.4