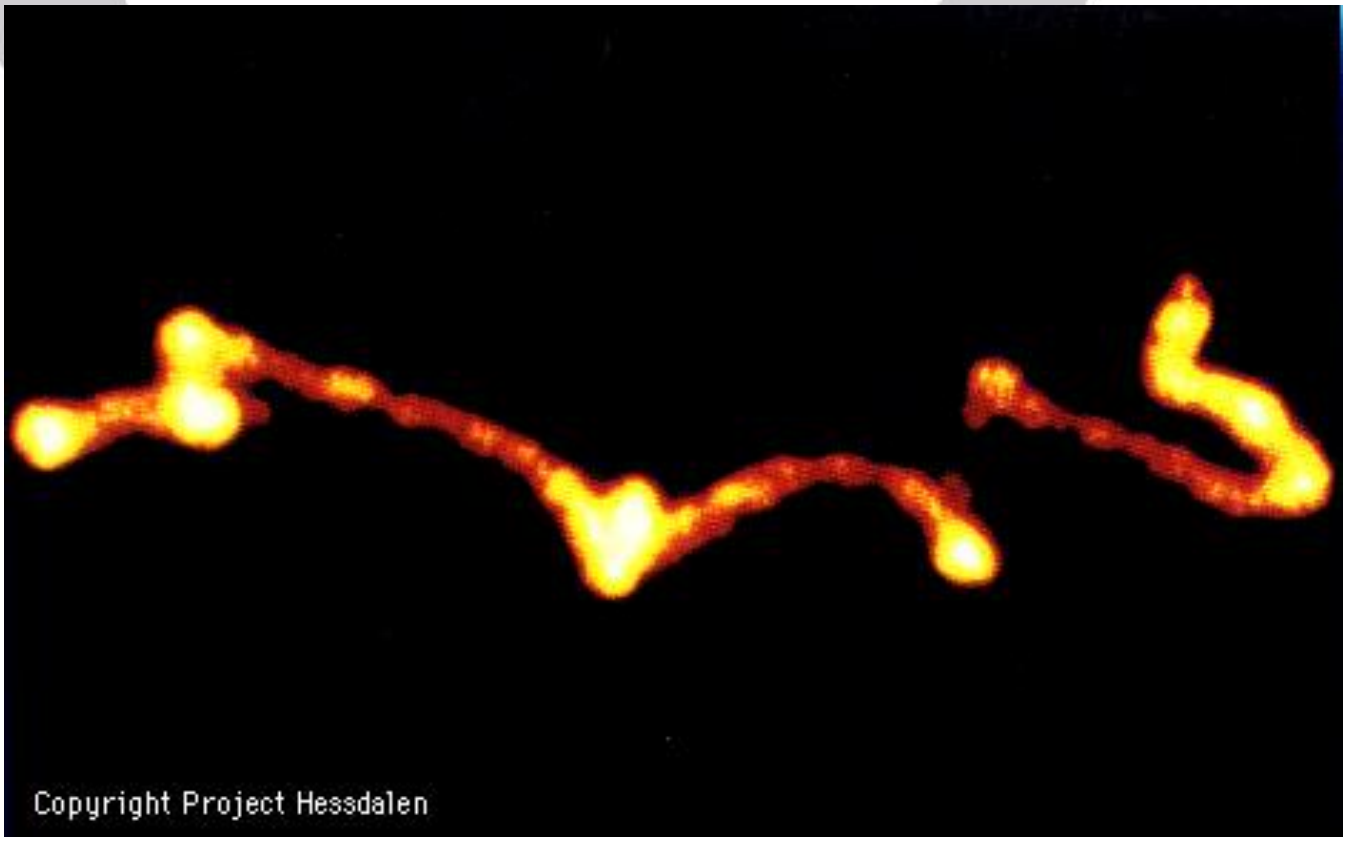
Hovedprosjekt 1998:

OBJEKT-DETEKTERING



av **Tommy Friberg**,
Øyvind Raaen og
Sverre Rekvin

HØGSKOLEN I ØSTFOLD

Avdeling for
informatikk og
automatisering

Postboks 1192, Valaskjold
1702 Sarpsborg
Telefon : 69 10 40 00
Telefaks : 69 10 40 02

Fritt tilgjengelig ☑

Tilgjengelig etter avtale med samarbeidspartner ☐

Oppgavens tittel:	**Dato:** 05.06.98
Objektdetektering	**Antall sider:** 27
	Vedleggsider: 39
Gruppedeltakere: Tommy Friberg, Øyvind Raaen, Sverre Rekvin	**Faglig veileder:** Martin Kermit
Avdeling: Avdeling for informatikk og automatisering	**Prosjektnummer:** A-1998-01

Utført i samarbeid med:	**Samarbeidspartners kontaktperson:**
CRULP	Erling P. Strand

Ekstrakt:

På oppdrag fra CRULP/Erling P. Strand, har vi utviklet en bildeanalyse som automatisk detekterer om det er kommet et ukjent, svevende objekt i Hessdalen. Løsningen på problemet baserer seg på å bruke et CCD-kamera i samkjør med en Silicon Graphics maskin, til å få et bilde vi analyserer ved hjelp av et eget C-program.

3 stikkord:

- CCD-kamera
- Bildeanalyse
- C-programmering

Forord

Som avslutningen av den tre-årige dataingeniørutdanningen, har vi hatt et 5 vekttalls hovedprosjekt de siste 2-3 månedene. Vi kunne velge mellom forskjellige prosjekter, men etter uttrekning og delvis valg fra studentenes side kom vi frem til at vi skulle ha dette prosjektet. Grunnen til at noen av oss valgte dette prosjektet, er interessen for det ukjente og ønsket om å få noen svar. Samtidig ønsker vi å få bruk for kunnskapene våre i praksis innenfor bildeanalyser, samt C-programmering, nettverk og UNIX.

Vi blir bedømt ut ifra kvaliteten på våre rapporter, kvaliteten på resultatet vårt og prosjektprosessen, inkludert EXPO. Derfor har vi brukt mye tid på å få en strukturert prosjektprosess. Dette har medført jevnlig rapportskriving, opprettholdelse av tidsplan og jevnlige oppdateringer på Internett.

I dette prosjektarbeidet føler vi at vi har hatt bruk for kunnskapene vi har lært i tidligere fag, kanskje spesielt "Synssystemer". Dessuten har prosjektet vært lærerikt innen flere felter, alt fra programmering til kunnskap om hva vi vet om ukjente lysfenomener, slik som de i Hessdalen.

Vi må benytte anledningen til å takke Erling P. Strand for innsikt, hjelp med prosjektets utføring og iver med å hjelpe oss til å få et bedre resultat på bildeanalysen. Og vi takker Marit T. Magnussen for korrektur.

Hjemmesiden vår: *http://sylfest.hiof.no/~a199801/*

Sverre Rekvin

Øyvind Raaen

Tommy Friberg

Innholdsfortegnelse

Sammendrag

CRULP, Center for Research on Unidentified Light Phenomena, er en forskningsgruppe ved HIØ. De har over lengre tid prøvd å finne ut av hva "UFO-fenomenet" i Hessdalen er. Fenomenet er observert ved veldig mange anledninger, og sees oftest som en kraftig lyskilde. For å skaffe mer informasjon og målbar data, skal det plasseres en automatisk målestasjon i Hessdalen.

Målestasjonen har blant annet et CCD-kamera. Et analyseprogram på en Silicon Graphics maskin bestemmer når videospilleren skal starte et opptak. Det er dette programmet vi skal lage ferdig. Programmet skal detektere ukjente objekter og helst luke ut normale fenomener, som fugler og billys.

Vi har ikke sett noe særlig på tidligere metoder å løse problemet på. Programmet er utviklet hovedsaklig fra "scratch", basert på egenlagde rutiner og prøving og feiling. Vi analyserer bildet vi får fra CCD-kameraet og luker ut generell støy. Der det er mye støy i bildet, har programmet en høyere terskel for å reagere på forandring i intensiteten i bildet. Når først programmet reagerer, lagrer det et bilde som skal legges ut på Internett, for at alle skal kunne se hva som skjer.

Det er lagd en konfigurasjonsfil for å kunne stille på forskjellige variabler for å få et optimalt resultat på objektdeteksjonen. Man er nødt til å stille seg frem avhengig av miljøet rundt og avstand og annet for å få et bra resultat. Stiller man det en vei, får man med altfor mye naturlig støy, mens hvis man stiller motsatt risikerer man å gå glipp av noen ukjente fenomener. Vi har også lagt til muligheten til å velge mellom å ta med bare lysende objekter, eller å ta med både lysfenomener og ukjente mørke flyvende objekter.

Denne sluttrapporten inneholder oppgaven vi fikk utdelt, metoden vi har brukt for å analysere bildet og hvordan andre skal kunne forandre programmet siden.

Innledning

Hessdalsfenomenet er et hittil ukjent fenomen. Det ses ofte som store lysende kuler av forskjellig form nær jordoverflaten. Lyset er fra noen centimeter stort til flere meter i diameter. Lyset er synlig i tidsrom fra under et sekund til flere timer. Lysstrålingen er intens og kan pulsere og skifte farge. Bevegelsene er uregelmessige. Fenomenet har ofte sfærisk eller elliptisk form. Det kan stå stille eller bevege seg inntil 30000 km i timen. I Hessdalen ved Røros kunne fenomenet observeres mange ganger ukentlig fra 1981 til 1985. Fenomenet er kjent fra en rekke andre steder i verden, dog ikke med samme hyppighet som i Hessdalen.

Hessdalen ligger i Holtålen kommune, omlag 3 mil nordvest for Røros. Selve dalen er omtrent 12 km lang og er omgitt av fjell på begge sider. Det bor omlag 150 mennesker der, og de fleste arbeider med jord- og skogbruk.

En gruppe forskere, inkludert Erling P. Strand dannet i 1983 Project Hessdalen. De foretok målinger av fenomenet i Hessdalen i to perioder i 1984-85. Mer enn 50 observasjoner ble gjort. Fotografier ble tatt, og disse viser fenomenets bevegelser og svingninger. Resultatene fra Hessdalen er blitt presentert på en rekke vitenskapelige konferanser, publisert som artikler, skrevet om i bøker og blitt omtalt i media. Fra hele verden har Hessdalsfenomenet blitt viet stor oppmerksomhet. En rekke anerkjente forskere har vist stor interesse for å gjøre inngående studier av fenomenet. En forskningskongress i Hessdalen i mars 1994 samlet 27 forskere fra 8 nasjoner.

Videre studier av fenomenet er viktig utfra flere perspektiver. Fenomenet er en ny og ukjent form for lys. Fenomenet representerer et hittil ukjent område innen fysikken. Videre studier vil kunne gi ny kunnskap. Disse omstendighetene førte til at CRULP ble dannet i 1993. Organisasjonen består av noen lærere ved Høgskolen i Østfold. Siden den gang har CRULP i de siste årene gjennomført en rekke studentprosjekter for å få satt opp en automatisk målestasjon i Hessdalen. Dette for å kunne undersøke fenomenet på en jevnlig basis, uten nødvendigvis å måtte dra dit opp. Containeren med målestasjon er tenkt å stå der oppe i en femårs periode.

Tidligere har det vært gitt en prosjektoppgave med formål å sette opp en UNIX-sentral i målestasjonen. Programmet på denne UNIX-sentralen inkluderte også en bildeanalyseringsrutine. Da denne rutinen ikke var god nok, fikk vi i oppgave å forbedre denne.

Oppgaven vår

Vi skulle stå for valget av et CCD-kamera som skulle erstatte Super VHS-videokameraet. Bildene fra kameraet blir sendt gjennom en bildegrabber på en SGI-maskin. Vi skulle utvikle rutinene som skal analysere bildene.

Programmet skal bare reagere på det ukjente Hessdalfenomenet. Det betyr at vi skal kunne skille ut støy, som værfenomener (snø, regn, hagl, fortløpende skyer og liknende), fugler, billys og huslys som slår seg av og på. Samtidig så skal ingen fenomener gå tapt. Derfor må vi heller ta med flere fugler enn å ta sjansen på å kutte ut noen ukjente fenomener. Best er det

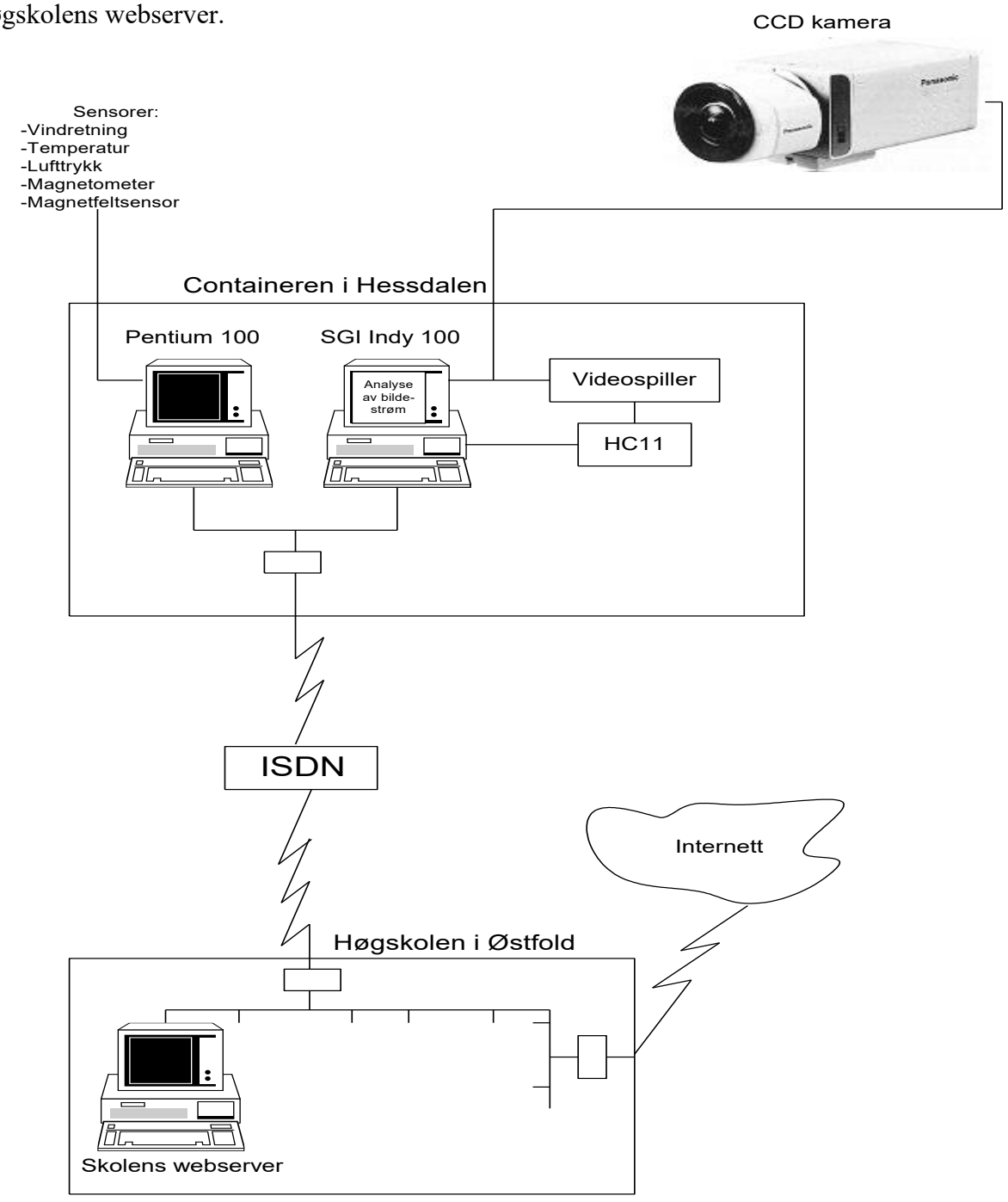
om vi får adskilt kjente og ukjente fenomener helt. Når programdelen reagerer, skal det sendes en beskjed til resten av målestasjonens program.

Som en uformell tilleggsoppgave som ble gitt under prosjektets gang, fikk vi i oppdrag å lage et Java-program. Det skulle vise frem en graf på Internett over temperaturmålinger gjort av den automatiske målestasjonen. Dette ble gjort, og javaprogrammet som viser trendkurver, er i operativ stand. Da dette er en uoffisiell oppgave, nevner vi ikke mer om den i denne rapporten.

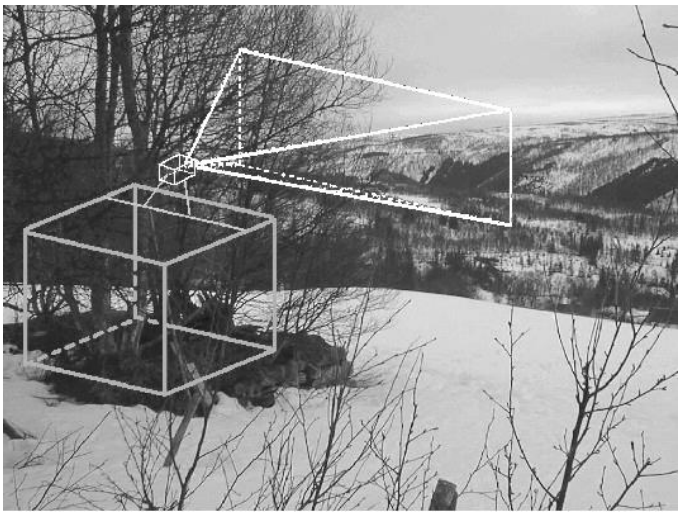
Utstyr og plassering

Den automatiske målestasjonen er satt opp i en container som skal beskytte datamaskiner og liknende utstyr mot vær og vind. Noen av sensorene står plassert utenfor containeren, noen skal blant annet stå montert på en grind på taket. Det skal være en strømkabel trukket opp til containeren, samt en ISDN-linje.

En Silicon Graphics Indy 100 kjører programmet som står for bildebehandlingen i prosjektet. Denne maskinen står i et lite lokalnettverk som er koblet opp til en ruter som igjen er koblet til ISDN-linjen. En Intel Pentium 100 er også koblet til dette nettverket. Denne PC'en står for en del andre målinger. Via ISDN kan da kontakt opprettes mellom høgskolen og containeren i Hessdalen. Høgskolen har fastlinje til Internett, slik at måledataene kan presenteres på høgskolens webserver.



Containeren huser den automatiske målestasjonen. Containeren, sett til høyre med Erling Strand foran, skal ha alt utstyret enten inne i seg, eller på taket. Den skal ha elektrisitet og ISDN-linje innebygd, slik at det bare er å plugge inn datamaskinene og annet utstyr.



Til venstre ser vi stedet hvor containeren mest sannsynligvis skal plasseres. Det er et gammelt stein-fundament i dalsiden, med utsikt utover resten av dalen. Vi ser her hvordan kameraet skal peke for å filme dalen.

Med vidvinkellinsa kan vi filme et stort stykke av dalen. Det bildet vi ser under, er cirka 180 grader horisontalt utover dalen. Rammen inne i bildet er et 107 graders utsnitt som kameraet fanger opp. (Vertikalt er det feil dimensjoner på bildet, da CCD-kameraet har 89 grader denne veien.) Kameraet skal plasseres litt høyere, oppe på containeren, i motsetning til da vi tok dette bildet, (hvor vi stod til knes i snø). Noen av trærne til venstre i bildet skal antakelig kappes ned, da de skaper for mye støy for filmingen.

Til høyre kan man se et kart over Hessdalen. Det er tegnet på synsvinkelen ut over dalen med basis i plasseringen av containeren. Vi kunne unngå kirken og veien ved å vinkle retningen på kameraet slik som på tegningen. Dette er ikke nødvendig i det man kan maskere bort deler av kamerabildet. Da vil ikke programmet trigge ved lysendringer i disse bortmaskerte delene av bildet.

CCD-kameraet (under) skal stå i et kamerahus på et stativ på containeren. Vi filmer Hessdalen med CCD-kameraet for å se etter ukjente fenomener. Dette kameraet har en vidvinkellinse, som gjør at vi kan filme med 107 graders horisontal vinkel. Den vertikale vinkelen er på 89 grader. Kameraet skal stå oppå containeren, på et stativ. En BNC-til-phono kabel skal være koblet mellom kameraet og en videospiller, som igjen er koblet videre til Indy-maskinen med SCART-til-phono kabel.

Før vi fikk CCD-kameraet, brukte vi et Super-VHS kamera. Dette ble brukt til å testfilme og undersøke hvor bra bildeanalyseprogrammet vårt fungerte. Vi filmet også Hessdalen med dette kameraet, slik at vi kunne få en oversikt over hvordan det så ut. Dette kameraet skal ikke være en del av målestasjonen i Hessdalen.

Det skal stå en Silicon Graphics Indy 100 i containeren. Et program som kjører på denne maskinen skal analysere bildene som kommer inn fra CCD-kameraet. Mesteparten av oppgaven dreier seg om denne bildeanalysen.

Det har vist seg at denne maskinen ikke er noe spesielt kraftig til bildebehandling på grunn av manglende prosessorkraft. Etter våre beregninger kan den i beste fall sammenlignes med en Intel Pentium 100.

Det går raskt å vise et bilde direkte fra framegrabberen til et vindu på skjermen, men programmet jobber for det meste med bildebuffere i RAM. Dermed hjelper det ikke at Indy-maskinen er rask til å grabbe bilder.

Hvis programmet på Indy-maskinen trigger, sendes det en beskjed til HC11-kortet om å starte opptaket på videospilleren (til høyre). Det er en vanlig VHS-maskin.

Dessverre tar det ca. 2 sekunder fra programmet trigger, til videospilleren begynner å ta opp. Dette er det ikke så mye å gjøre med, da videospilleren trenger denne tiden fra den får signal om å starte, til den begynner å ta opp.

I tillegg til dette utstyret, har vi en ruter, en pentium 100, HC11-kort, masse kabler (bl.a. nettverksutstyr) og litt faglitteratur (se kilder). Ruteren sørger for at maskinene på lokalnettet får tilgang til ISDN-linjen. Pentium-maskinen brukes til å prosessere måledata fra de andre sensorene, nemlig vindretning-, temperatur-, lufttrykk- og magnetfeltssensorer, i tillegg til et magnetometer. HC11-kortet brukes til å starte opptaket på videospilleren på beskjed fra detekteringsprogrammet som kjører på Indy-maskinen. Disse to kommuniserer via vanlig RS232-interface. Fagmaterialene har sjeldent vært til noen hjelp, siden mesteparten av programmeringen er basert på egenskapte rutiner.

Vi har også lånt tre pentiummaskiner av HIØ, samt en 14" TV. Disse har vi brukt til å skrive rapporter, lage webgrensesnitt, teste bildene fra kameraet og liknende. Utstyret skal leveres tilbake etter endt prosjekt.

Programmet

Det er lagd et program for den automatiske målestasjonen, som kommuniserer med alle sensorer på målestasjonen (lagd ferdig av Martin R. Thorsen). Vi tok ut delen av denne som hadde med bildeanalysen å gjøre. Vi gikk ut ifra verdiene som gikk ut og inn av den delen. Så lagde vi et programskall som sendte inn verdier og hentet verdier tilsvarende hovedprogrammet.

Den gamle bildeanalysedelen baserte seg på kode som var kjapt skrevet som en deloppgave av et prosjekt. Grunnlaget var feil for oss å fortsette på. Derfor lagde vi en helt ny rutine for å analysere bildet som kom inn fra kameraet.

Filene

Vi har en *main.c*, som et programskall. Programdelen *analyze.c* og *analyze.h* skal analysere bildet som kommer inn fra videograbberen og finne ut om videospilleren skal begynne å ta opp fordi det er et ukjent fenomen som forekommer. Dette er det programmet som tar seg av all bildeanalysen, som hovedsaklig er det prosjektet dreier seg om. I tillegg til disse to programdelene, må det være en *grab.c* og en *grab.h* også, som kun tar seg av å hente bildet fra videograbberen. Denne programdelen har vi hentet ned fra SGI sitt nettbibliotek og endret litt på.

Det er lagt inn et program på Indy-maskinen, *convert*, som gjør om et bilde fra rå *.RGB*-fil over til *.JPG*-fil eller *.TIFF*-fil. Programmet brukes av *analyze.c* for å gjøre om *.RGB*-filene skapt av *grab.c*, til filer forstått av andre vanlige brukte bildebehandlingsprogrammer.

Når vi kompilerer programmet (vha. unix' *make*), får vi en kjørbar fil som heter *ana*.

Det benyttes en konfigurasjonsfil, *a199801.conf*, for å kunne endre på programmets oppsett uten å måtte rekompilere programmet. Denne ligger i samme katalog som programfilene.

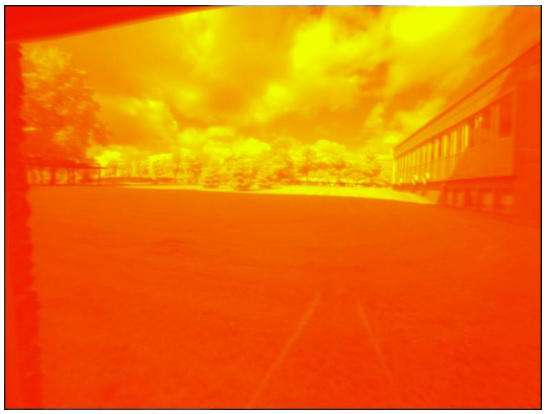
Det blir også skrevet til en logfil i *analyze.c*, *a199801.log*. Denne er for at vi skal kunne se når det har blitt detektert noe med programmet. Denne ligger også/skal også ligge i samme katalog som programfilene.

analyze.c, *analyze.h*, *grab.c* og *grab.h* skal settes inn i Martins program. Deretter må programdelene forandres slik at de tilpasses hele programmet. Konfigurasjonsfilen må også inn i katalogen hvor programmet ligger, hvis ikke henvisningen til filen forandres i *analyze.c*. Logfilen er til for at vi skal kunne teste programmet riktig. Det vil nok være en rutine i hovedprogrammet til Martin som setter opp en bedre logfil enn den vi har brukt i prosjektperioden.

Hvordan fungerer programmet?

I vårt tilfelle startes programmet med filen *ana*. Hvis det er satt opp i konfigurasjonsfilen, *visvindu 1*, vil et vindu komme opp, med kameraets bilde.

Figur A



Det første bildet er farget med gult og rødt. Dette kommer av at programmet må kalibrere seg.

Figur B

Det gule/røde vil avta. Når dette er ferdig, vil det være igjen et svart/hvitt bilde av det kameraet ser.

Figur C

Når et ukjent fenomen detekteres vil det merkes ved at området som trigget programmet er merket med gul-rødt. (Se billysene som er blitt tent.)

Selve analysen som er for å finne ut hva programmet skal trigge på, fungerer på følgende måte:

- Vi finner **differansebildet** mellom **gjennomsnittsbilde** og nåværende bilde. (Gjennomsnittsbildet starter med å være helt svart, så i starten får differanse-bildet høy intensitet.)

- Et nytt **gjennomsnittsbilde** blir kalkulert.

- Vi oppdaterer lokasjon av **støy i bildet**. Disse stedene blir regnet som støyområder, og skal ikke være så sensitive på å oppdage ukjente fenomener.

- Vi finner ut hvilke punkter i bildet som har høy intensitet, som programmet skal reagere på. Terskelen for at programmet skal trigge blir bestemt ut ifra støybildet.

Differansebildet

Differansebildet er en tabell som blir generert slik:

$$Differansebildet = |Nåværende\ bilde\ fra\ grabberen - Gjennomsnittsbildet|$$

Som nevnt over, starter gjennomsnittsbildet med å være et helt svart bilde med null intensitet. Da får differansebildet høye verdier til å begynne med.

Det blir tatt absoluttverdien av differansen på høyre side, hvis konfigurasjonsfilen er stilt inn på å ta både lyse og mørke fenomener, *kunlyse 0*. Hvis det nåværende bildet har lavere intensitet (mørkere) enn gjennomsnittsbildet, blir det et negativt tall ut av subtraksjonen. Dette blir da regnet som en positiv forandring i differansebildet på grunn av absoluttverdien.

Hvis man bare skal detektere lyse objekter, *kunlyse 1*, ser man bort fra absoluttverdifunksjonen, slik at negative tall fra subtraksjonen blir regnet som ingen forandring i differansebildet. Dette blir vel og merke bare forandret på under deteksjon.

Eksempel
kunlyse er satt til *1* i konfigurasjonsfila.

nåværende bilde hentet fra grabberen som skal undersøkes

23	25	23	25	24
26	27	20	26	37
23	66	57	50	24
23	35	47	60	25
24	23	27	26	28

gjennomsnittsbildet som er blitt bygd opp i løpet av lengre tid

23	25	23	25	24
26	27	24	30	35
23	26	28	30	24
23	35	27	26	25
24	23	27	26	28

differansebildet som et resultat av formelen over

0	0	0	0	0
0	0	4*	4*	2
0	40	29	20	0
0	0	20	34	0
0	0	0	0	0

*) egentlig negative tall, men blir satt til positive i differansebildet på grunn av absoluttverdi, men disse tallene blir regnet som negative under deteksjon, det vil si ingen verdi (se under)

Gjennomsnittsbildet

Gjennomsnittsbildet er ikke fullstendig gjennomsnittet av de forrige bildene, så det er litt feil å kalle det dette. Vi regner bare på ett bildebuffer for gjennomsnittsbildet, for å spare maskinkraft. Det er en tabell der hver piksel har fått tildelt 32 bit. De 8 øverste bitene er ubrukte. Bit 0-23 er intensiteten til punktet. Hvis en skal konverter dette til et 256 gråskala er det bare å hente ut bitene 16-23. Hvert punkt har fått intensiteten beskrevet 24 bit, fordi vi ønsker å bruke tall med desimaler. Vi har valgt dette formatet for å unngå bruk av flyttall, for å spare maskinkraft.

Oppdatering av gjennomsnittsbilde, skjer på denne måten:

Endring = Nåværende bilde - Gjennomsnittsbildet

*Gjennomsnittsbildet = Gjennomsnittsbildet + Endring * gjennomsnittsoppdatering*

Oppdateringsfaktoren er et tall mellom 0 og 1, som blir stilt i konfigurasjonsfilen. Dersom verdien er 0, vil ikke gjennomsnittsbildet bli oppdatert. Hvis verdien er 1 vil det nåværende bildet bli gjennomsnittsbildet. Hensikten med å sette opp et gjennomsnittsbilde er å fjerne støy, og å kunne justere hvor mye forandring i bildet den skal kunne reagere på. Oppdateringsfaktoren bør være litt over 0.

Eksempel
gjennomsnittsoppdatering er satt til 0.2 i konfigurasjonsfila. Vi bruker *Nåværende bilde* og *Gjennomsnittsbildet* fra forrige eksempel. Forandringen i gjennomsnittsbildet er disse to subtrahert ganget med 0.15.

Endring regnet fra *Nåværende bilde - Gjennomsnittsbildet*

0	0	0	0	0
0	0	-4	-4	2
0	40	29	20	0
0	0	20	34	0
0	0	0	0	0

Endring ferdig ganget med *gjennomsnitts-oppdatering*

0	0	0	0	0
0	0	-0.8	-0.8	0.4
0	8	5.8	4	0
0	0	4	6.8	0
0	0	0	0	0

Det nye *gjennom-snittsbildet* som et resultat av formelen

23	25	23	25	24
26	27	23.2	29.8	35.4
23	34	33.8	34	24
23	35	31	34.8	25
24	23	27	26	28

Støybildet

Støybildet er en tabell der hvert punkt i bildet har fått hver sin verdi som angir støy i bildet. Hvis verdien i differansebildet er større enn støybildet, vil støyverdien i punktet øke med en ”Læringsverdi”, *stoylaering*. Læringsverdien er til for at trær, skyer, måne, sol, regn og snø ikke skal detekteres. Hvis Støyverdien til et punkt er mindre enn differansen til punktet, vil støyverdien minske med ”Glemmeverdi”, *stoyglemming*. Vi får:

Støyverdi = Støyverdi + Læringsverdi

Eller

Støyverdi = Støyverdi - Glemmeverdi

Eksempel
stoylaering og *stoyglemming* er satt til respektive *3.5* og *0.5*. Hvis *differansebildet* er større enn *støybildet*, så øker *Støybildet* med *stoylaering*. Motsatt minsker *Støybildet* med *stoyglemming*.

Differansebildet

0	0	0	0	0
0	0	4*	4*	2
0	40	29	20	0
0	0	20	34	0
0	0	0	0	0

Støybildet

3	2	0	1	1
2	5	2	3	4
4	2	5	6	6
5	4	4	6	2
0	4	6	1	7

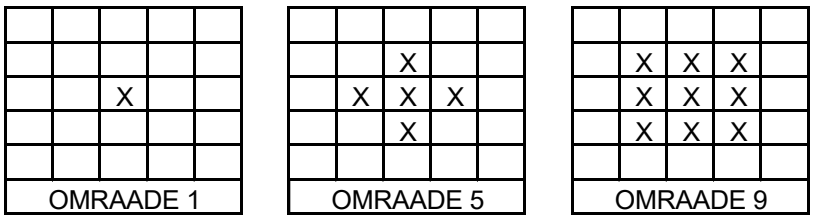
Det nye *Støybildet*

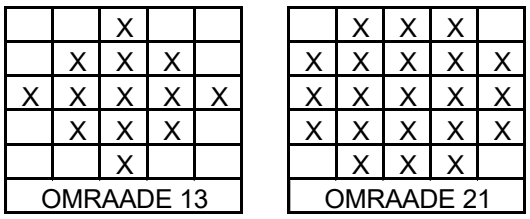
2.5	1.5	0	0.5	0.5
1.5	4.5	5.5	6.5	3.5
3.5	5.5	8.5	9.5	5.5
4.5	3.5	7.5	9.5	1.5
0	3.5	5.5	0.5	6.5

*) på støyforandringsrutinen blir negative tall regnet som positive

Deteksjonen

Programmet vårt kan kjøres i 5 forskjellige modus, *omraade 1*, *5*, *9*, *13* og *21*. Hver pixel i bildet blir sjekket. I modus 1 blir det kun tatt hensyn til støyen (hentet fra Støybildet) i ett punkt. I modus 5 blir støyen hentet fra fire punkter rundt punktet og punktet selv. Her kan man se hvordan det blir hentet punkter rundt midtpunktet ved de forskjellige modus:





Vi adderer sammen støyen i punktene og dividerer med en verdi, *omraadedivisjon*. Dette resultatet pluss en *minimumdifferanse* sammenlignes med Differansebildet. Så teller vi opp antall punkter i bildet som vi har fått deteksjon på. Hvis antall punkter ligger mellom *minantallpunkter* og *maksantallpunkter*, trigger programmet ved å lagre bildet den trigget på og skrive i logfilen.

Eksempel
omraade er satt til *13*. *omraadedivisjon* er satt til *3.5*. *minimumsdifferanse* er satt til *10*. *minantallpunkter* er satt til *5*. *maksantallpunkter* er satt til *10000*. Siden dette er et lokalt eksempel, teller ikke antall punkter. Vi samler støyen fra de punktene som er merket med '.

Støybildet

3	2	0'	1	1
2	5'	2'	3'	4
4'	2'	5'	6'	6'
5	4'	4'	6'	2
0	4	6'	1	7

Utregninger

sum av punktene: 53

sum / omraadedivisjon: 15.14

lagt til minimums-differanse: 25.14

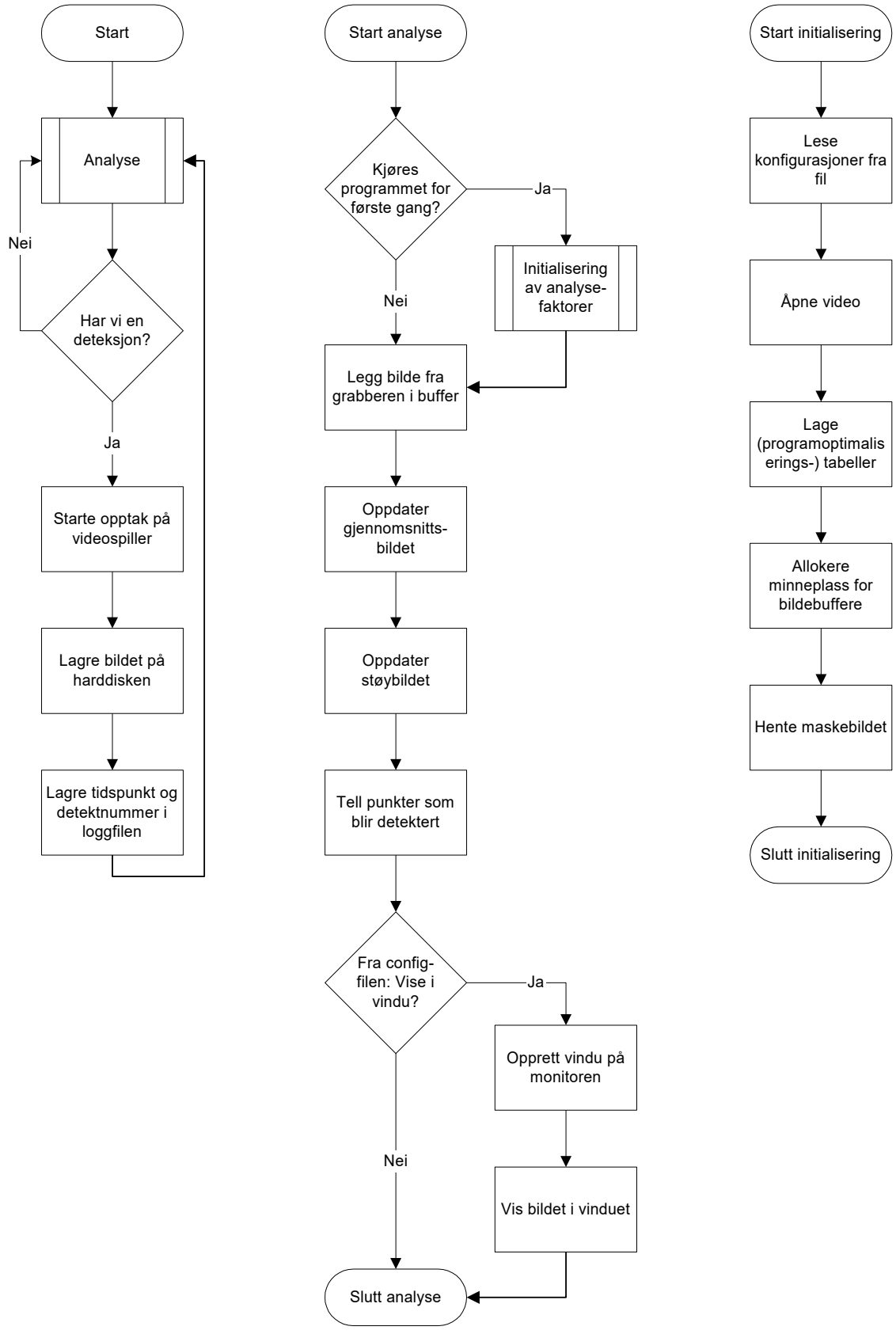
Differansebildet

0	0	0	0	0
0	0	-4*	-4*	2
0	40	29!	20	0
0	0	20	34	0
0	0	0	0	0

*) de negative tallene i *Nåværende bilde - Gjennomsnittsbildet*

!) Vi ser at det midtre punktet i differansebildet er større enn utregningen av støyet i området. Dette vil da gi en triggring av programmet.

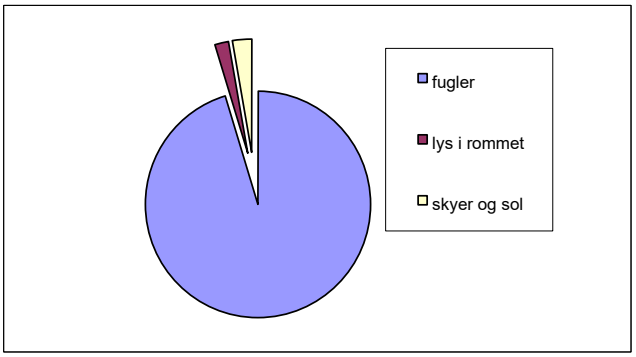
Blokkdiagram over programmet



Tester

Vi har kjørt mange tester på programmet. Den første testen ble gjennomført fra den 4.5.1997 kl 1700 til den 5.5.1997 kl 1030 . Vi hadde glemt å skru av autofokus, derfor ble denne testen mislykket. Mellom lyse og mørke bilder vil kameraet fokusere og da reagerer programmet vi brukte da. Det vi kunne se ut fra denne testen var de fleste deteksjonene kom av fugler, og arten måker var veldig godt representert.

Den andre testen 5.5.1997 kl 1700 til 6.5.1997 kl 1230. I den testen fikk vi følgene resultat:



Fugler:	102
Lys i rommet:	2 (Refleksjon fra vinduet foran kameraet)
Skyer og sol:	3
Totalt	107

Vær: Lite vind. Sol i begynnelsen og resten overskyet.

Lys i rommet er lys på rom U59 som blir slått av eller på, dette påvirker kameraet fordi lyset i taket speiler seg i vinduet som vi filmer gjennom. Skyer og sol er værfenomener som kameraet har reagert på.

I den tredje testen fikk vi 65 fugler av 67 registreringer. De to resterende var lys eller vindu som ble åpnet. Vi hadde her ingen værforstyrrelser.

Vær: Lite vind, overskyet.

Eksempel på måke:

Dette bildet er representativt for de fuglebildene som programmet vårt har registrert. Vi ser at det skal ikke til store området før programmet detekterer noe.

Etter disse testene, gjorde vi om programmet slik at vi bare detekterte lyse objekter. Slik unngikk vi de aller fleste fugleproblemene. Testene som fulgte etter dette, var veldig suksessfulle. De bare reagerte på billys og andre lyseffekter skapt av blant annet solen. Det har skjedd adskillige ganger at vi har reagert på støy fra trær og liknende naturlig støy, men dette er fordi konfigurasjonsfilen har vært dårlig satt opp i forhold til miljøet rundt.

Ting vi har prøvd på, men forkastet, og hvorfor

Vi har funnet ut at skarpe overganger mellom lyse og mørke objekter skaper støy, som igjen kan skape deteksjoner. Derfor har vi innført støybildet. En mulig løsning på dette var å kjøre en kant-deteksjon på bildet, og lage en detekteringsfunksjon som reagerte dårligere der det var en kant i bildet. Vi har kastet denne idéen, fordi det tok for mye prosessortid. Vi fant ut at i skarpe overganger vil det hele tiden komme støy, derfor var det enda en god grunn til å forkaste denne. I ettertid, med det nye CCD-kameraet, ble det mindre støy generelt på hele bildet, så vi klarte oss med innføringen av støybildet.

Vi fant ut at programmet reagerte på trær som står nærmere enn ti - femten meter. De lager støy med greinene. Vi tenkte ut idéer for å fikse dette. Da vi kom til Hessdalen, fant vi ut at det ikke var noe problem. Vi har bestemt oss for at trær som står nærmere enn ti meter, blir antakelig kappet, eller at de blir merket bort i dataprogrammet, (dog blir da et mindre område enn nødvendig å detektere på). Dessuten kan man stille i konfig-filen følsomheten på dette området, slik at støyverdien fra greinene på trærne forblir høy hele tiden, slik at falske deteksjoner unngås.

Vi skulle finne ut om vi skulle analysere bildene i farger eller i svart/hvit. Grunnet prosessorkraft, og at vi ikke fant det noe hensikt i å bruke fargeinformasjonen til noe, har vi valgt svart/hvitt. Det tar nemlig 3 ganger så lang tid å analysere et bilde i farge enn i svart/ hvitt. Dessuten har vi valgt et svart/hvitt CCD-kamera. Vi valgte dette, fordi denne typen kamera har langt høyere lysfølsomhet en fargekameraer av samme typen. Panasonic-kameraet som vi har kjøpt, har en lysfølsomhet på cirka 0.02 lux, 50 ganger mer lysfølsomt enn Hi8-kameraet vi testet med i starten av prosjektperioden. Dette Hi8-kameraet var også regnet for å være relativt bra.

Vi hadde også et problem med at programmet detekterte for bra, det vil si at det trigget på naturlige flygende fenomener, f.eks fugler, flaggermus og insekter. Det ble foreslått at vi skulle prøve å lage en kantdeteksjonsrutine på det området den trigget på, for å se om vi kunne gjenkjenne visse fenomen. Vi erfarte det at selv menneskeøyet klarte bare å stadfeste at cirka 10% av deteksjonene var helt sikkert fugler. Det er ikke unaturlig å tenke seg at programmet bare hadde klart å gjenkjenne en liten del av disse 10%, dermed ble det kanskje bare 1% av fugledeteksjonene som var mulig å skille ut ved å implementere en slik rutine. Dessuten var det fremdeles problemer med at dette ville ta mye prosessorkraft, og dermed ville oppfriskningsraten falle betraktelig. Dette var ikke ønskelig, så vi forkastet også denne ideen.

Hvordan jobbe videre på det vi har?

Vi mener at produktet vårt nå er ganske komplett nå. Selvsagt er det mulig å forbedre dette, og også fysisk fortsette på kildekoden, men inntil man får en bedre maskin, ser vi ikke noen vits i å endre noe i kildekoden. Programmet er lagt opp på en slik måte at det er mulig å endre på veldig mye via konfig-filen, slik at så få endringer som mulig skal måtte implementeres direkte i kildekoden.

Prinsippene bak programmet og kildekoden i seg selv er godt forklart, så det ligger til rette for at noen kan fortsette på koden. Erfaringsmessig ser man at når man skal fortsette på en produktidé, ser man vanligvis bare på prinsippene bak programmet, og lager alt fra bunnen av allikevel. På den måten får man et mest mulig effektivt og feilfritt program.

Av andre endringer kan vi nevne at et enda bedre kamera også ville gitt bedre resultater. Dersom man for eksempel hadde hatt et kamera med meget høy oppløsning, og en grabber med tilsvarende oppløsning, kunne man satt på en veldig stor vidvinkel, og allikevel fått nok informasjon på bildet til å kunne trigge på eventuelle fenomener. Større vidvinkel innebærer også at kameraet burde være enda mer lysfølsomt. Vi har undersøkt slike kameraer, men de koster altfor mye, i forhold til budsjettet til Project Hessdalen på dette stadiet.

Vi mener også at med en mye kraftigere maskin, kunne man ha implementert mye kraftigere bildebehandlingsrutiner, men man ville antakelig ikke tjent så veldig mye på det, med hensyn til eliminering av falske deteksjoner. Slik det er nå, med tanke på maskinkraft og implementerte funksjoner, mener vi at det fungerer meget tilfredsstillende.

Endringer i programmet som vi kunne ha gjort med mer prosessorkraft

Oppdatering av støyen i bildet skjer nå lineært. Vi hadde ønsket å sette opp en PID-regulator for hvert punkt. Dette tror vi kunne gi bedre kontroll, slik at det hadde vært lettere å stille inn programmet og fått bedre resultater med tanke på deteksjon.

Oppdragsgiver ønsket seg en metode for å kunne detektere størrelsen på objekter. Programmet vårt gjør ikke det på dette tidspunkt. Den gir kun antall punkter i bildet som er detektert. Dette er fordi at når programmet vårt detekterer et objekt, er nesten alltid punktene samlet.
Hvis vi hadde hatt en rutine som detekterte størrelsen på et objekt, kunne det ha vært interessant å sette opp følsomheten på programmet. Da ville vi antakelig fått mange ensomme deteksjoner spredd rundt på bildet. Disse punktene skal da ikke bli regnet med, fordi de ikke er samlet til et objekt. Da oppstår et problem: I ca 50% av tilfellene er objektet som vi ønsker å detektere, delt i flere deler. Da må man endre objektstørrelsen for deteksjon slik at den ikke bare detekterer punkt i det nærmeste område rundt objektet, men også tar punkter lenger unna rundt objektet. Vi måtte også implementere en funksjon som ikke starter en deteksjonsprosess rundt et ensomt detektert punkt. Dersom vi skulle gjennomføre dette, måtte vi gå ned på antall pixler vi tester på, det vil også si at størrelsen på et fenomen som skal bli detektert måtte bli større. Derfor har vi valgt å ta det ut, da det var vesentlig at å detektere små objekter. Hvis vi hadde hatt mere prosessorkraft, ville vi snarere ha brukt denne til oftere oppdatering av bildene.

Vi har ikke undersøkt så veldig mye om mer avanserte metoder for å detektere objekt, fordi maskinen som sagt har for liten prosessorkraft. Til videre utvikling av dette prosjektet anbefaler vi en oppgradering av eksisterende hardware.

Planer og rammer under prosjektets gang

Vi satte opp en ramme på hva vi skulle være ferdig med til bestemte tider. Denne rammen har vi holdt, bortsett fra når det gjelder skriving av rapporter og oppdatering på nettet. Systemutviklingsrapporten ble forsinket blant annet på grunn av trøbbel med Windows NT og Word 97. Dessuten var personen som var ansvarlig for denne delen opptatt med å lese til en matematikkeksamen. Hovedgrunnen til drøyingen av leveringen var kvalitetssikring av rapporten. Systemutviklingsrapportens oppsett var det ment å bruke under skrivingen av sluttrapporten også. Derfor måtte den være relativt omstendig.

Når det gjelder prosjektets resultater, mener vi at vi har klart de mål vi satte oss på forhånd. Vi satte ikke opp noen spesielle delmål på programmeringsdelen, da vi på dette tidspunktet ikke visste hva programmeringen kom til å inneholde i detalj. Vi satte aldri opp noen bestemte mål for når ting skulle være ferdig på programmeringen. Vi erfarte raskt det at vi hadde god tid til å gjennomføre den gitte oppgaven, og fikk ganske tidlig sammen et brukbart fundament. Dermed var det mest testing og kalibrering som gjenstod. Ting som loggfil, konfig-fil og lagring av bildet på harddisken er eksempler på finpuss, som egentlig ikke tar så lang tid å gjennomføre, og dermed ikke trengte mye planlegging.

Internett-delen har vært rimelig godt dekket, til tross for dårlig tid. Det er mye mer vi kunne tenkt oss å forbedre på denne delen, men vi har ikke nok arbeidstimer til overs. Dette har vist seg å være en heltidsjobb å oppdatere alt og se til at alt fungerer. Dette har vi delt litt på, siden ingen har hatt tid til å bare drive med dette. Ting vi ikke har rukket å gjennomføre på det nåværende tidspunkt, er blant annet en engelsk versjon av websidene, samt en liten del som viser oppgaven og løsningen i detalj. Alle rapporter, samt denne sluttrapporten, er på websidene. Vi mente at det var viktig med gode websider, da vi vet at mange mennesker er interessert i både Hessdalen, Project Hessdalen og målestasjonen, og bruker Internett som informasjonskilde. Vi har nå ordnet det slik at man kan søke på enten Hessdalen eller objektdetektering på en søkemaskin (f.eks. Altavista), så vil man få opp våre sider. Vi vil prøve å gjøre Internettdelen helt ferdige etter prosjektlevering.

Når det gjelder de tidlige fremdriftsplanene, så har vi med et lite slingringsmonn holdt fristene, og nå gjenstår hovedsaklig gjennomføringen av EXPO. Vi har allerede planlagt noe, slik som plasseringen av utstyret på EXPO, litt om foredrag, og avtalt med gruppa som utstyrer containeren.

Generelt har samarbeidet mellom oss fungert bra, og arbeidsoppgavene har blitt fordelt uten problemer. Vi har kanskje vært heldige som har klart oss så godt uten administrering av oppgavene og systematisering av tidsfrister og liknende. For eksempel, så har ingen vært leder i gruppa. Vi har alltid diskutert og kommet frem til enighet på alt som har vært verdt å diskutere. Vi har tatt en avgjørelse fra starten av at vi ikke trengte en leder, og vi har bestemt oss for at vi klarer oss uten en altfor detaljert fremgangsplan. Ved å slippe å bruke tid på dette, har vi hatt mer tid på andre, mer nødvendige oppgaver. Vi er bare tre, og dette har fungert glimrende så langt. Hvis vi derimot hadde vært flere, eller oppgavene hadde vært mer detaljerte fra starten av, ville vi nok brukt en annen fremgangsmetode.

Konklusjon

I forhold til oppgaven som er gitt, mener vi at vi har fått et bra resultat. Programmet tar hensyn til støy, og dersom det er riktig innstilt, vil det ikke detektere fenomener forårsaket av støy.

Sett bort i fra at det antakelig er færre fugler i Hessdalen enn i Sarpsborg, så har vi nå implementert en løsning på problemene med falske deteksjoner forårsaket av fugler. Ved testing har vi funnet ut at ved å bare trigge på lyse fenomener, vil nærmest alle fuglebildene forsvinne. Dette medfører at en del lysfenomener som skjer i dagslys nok ikke vil bli oppdaget, men dette gjenstår å se. Det vil også variere ettersom hvor sterkt fenomenet stråler.

Vi har også implementert "maskebilde-prinsippet". Dette innebærer at man kan grabbe et enkelt bilde, og markere bort områder som programmet ikke skal detektere noe i. Det eneste problemet med at kameraet nå kan siktes mot områder med lysaktivitet, er at selve kameraet justerer lysintensiteten i bildet etter en senterjusteringsmetode. Dermed kan bildet bli for mørkt, (som medfører at man ikke får noen referanser i bildet), dersom det er et skarpt lys nær sentrum av bildet. Dette må den som plasserer kameraet ha i bakhodet, og teste ut grundig.

Vi valgte å kjøpe et svart/hvit CCD-kamera fra Panasonic (wv-bp310) med en høyhastighetslinse (High-Speed Aperture Lenses) med vidvinkel 107°. Det vi har rukket å teste med dette kameraet, har gitt meget positive resultater med hensyn til kameravinkel og lysfølsomhet. Vi har også kjøpt inn et kamerahus (ip-66 standard) med termostat, slik at kameraet skal kunne operere hele året, uten problemer.

Vi har erfart at det trengs en raskere maskin til bildebehandlingsprosessen. Den vi har, bruker for lang tid på alle prosessene sammenlagt. Dersom man hadde hatt mer prosessorkraft, kunne man først og fremst hatt en raskere oppdatering, i tillegg til at vi kunne implementert tyngre algoritmer for deteksjon, (dog mener vi at det vi har nå, fungerer bra nok uansett). Men med en raskere oppdatering kunne vi blant annet analysert banen til objektene, og kanskje fjernet noen falske deteksjoner på denne måten.

Da vi bruker relativt få, enkle og optimaliserte prosesser på bildet, klarer vi altså å sjekke bildet for fenomener med cirka et sekunds mellomrom. Resultatene blir lagret på harddisk, og det blir lagret tidspunkt (i en loggfil), deteksjonsnummer og antallet punkter som er detektert.

Kilder

Bøker

Kochan, Stephen G. (1994), *Programming in ANSI C, revised edition*, SAMS

Spainhour, Stephen; Quercia, Valerie (1996), *Webmaster In A Nutshell, A Desktop Quick Reference*, O'Reilly

Akeley, Kurt m.fl. (1992), *OpenGL Reference Manual*, Addison-Wesley, 4. opplag 1995

Karlsen, Terje m.fl. (1990), *Kommunikasjon for ingeniører*, Universitetsforlaget, 2. opplag 1992

Havik, Leif (1987), *UFO-fenomenet - Kan det umulige være mulig?*, Vision

Log, Steffen (1997), *Vedleggsdeler til faget synssystemer*, Høgskolen i Østfold

Flanagan, David (1996), *Java In A Nutshell, A Desktop Quick Reference*, O'Reilly

Andresen, Sissel Authen; Jacobsen, Karsten; Stadskleiv, Einar (1990), *Prosjekthåndbok med retningslinjer for studentprosjektvirksomheten ved Østfold Ingeniørhøgskole*, Østfold Ingeniørhøgskole, 3. opplag 1995

Internett

- http://techpubs.sgi.com/library/, april-mai 1998

- http://www.hiof.no/crulp/, april-mai 1998

- Usenet (news/diskusjonsgrupper på nettet), april-mai 1998

Diverse

- Diverse filmopptak av Erling Petter Strand av Hessdalenekskursjoner, datert ca. 1984-85

- En demo på programmet *Thomas Register of European Manufacturers*, 1998

Andre relevante prosjektgrupper ansatt av CRULP:

1998 - E-1998-04: Automatisk Målestasjon

1997 - A-1997-05: UNIX-Sentral i Målestasjon

- E-1997-05: Automatisk Målesystem; Hessdalen

1996 - A-96-01: UNIX-Sentral i Målestasjon

1994 - E-94-01: Radiopeilesystem

- E-94-04: Automatisk Målestasjon

- E-94-05: Videofølgesystem

Programkoden (vedlegg)

analyze.c
Vi har valgt å konsentrere oss om å forklare koden i *analyze.c*, siden det er denne som er kjernen i prosjektoppgaven.

```
#include <stdlib.h>
#include <stdio.h>
#include <gl/gl.h>
#include <sys/types.h>
#include <time.h>

#include <dmedia/vl.h>
#include "grab.h"
#include "analyze.h"

/*program instillinger som hentes fra logfil*/


static long gjennomsnittsoppdatering;/*hvor raskt bilde skal oppdatere
seg*/
static long omraade;                 /*Hvor stort omraade */
static long omraadedivisjon;         /*omraade sensitivitet. samletstoy /
                                       omraadedivisjon*/

static long minimumdifferanse;       /*minimum differanse*/

static long stoylaering;             /*laere faktor*/
static long stoyglemming;            /*glemme faktor*/
static long minantallpunkter;        /*min for at den skal trigge*/
static long maksantallpunkter;       /*max for at den skal trigge*/
static long kaliberingsbilde;        /*hvormange bilder en skal ta for den
skal
                                     kunne ha mulighet til aa kunne
trigge*/
static long visvindu;                /* 0 = ikke vis vindu */
static long kunlyse;                 /*1=hvis en kun skal finne lysere
                                       objekter*/
static long brukemaske;              /*1+hvis bruke maske*/

/* ikke program instillinger */



static int openvideoch=0;          /*test verdi for video grabber er
                                     initalisert*/
static int gjennomsnittsbildech=0; /*test verdi for buffere generert 0/1 */


/*peker til buffer som programmet trenger */
static long *gjennomsnittsbilde;   /*gjennomsnittsbilde*/
static long *differanse;           /*differansebildet */
static long *stoy;                 /*stoybildet*/
static char *funnet;               /*hvilket punket som har blitt funnet
0=ikke funnet 0xff+funnet*/
static long *winbuffer;            /*bildet som visespaa skjermen*/
static char *mask;                 /*maske bildet*/
```

```
static long antallfunnet;          /*antall piksler funnet i bildet*/
static long behandlet;             /*antall bilder behandlet siden siste
deteksjon */

static  long win;                  /*vindu id*/
static  int xstorrelse;            /*x storrelse paa bildet fra grabberen*/
static  int ystorrelse;            /*x storrelse paa bildet fra grabberen*/
static  char *datafragrabber;      /*peker som blir bruke til aa peke paa
data fra grabberen*/


static long stoyfilter[]=
{
  0,0,

  0,1,
  0,-1,
  1,0,
  -1,0,

  -1,1,
  1,1,
  1,-1,
  -1,-1,

  0,2,
  2,0,
  0,-2,
  -2,0,

  -1,2,
  1,2,
  2,1,
  2,-1,
  1,-2,
  -1,-2,
  -2,-1,
  -2,1,
};
static long stoyfi[21];
/* stoyfilter hvilke pikler i omraadet rundt som en skal ta hendsyn til.
   stoyfi blir generert av stoyfilter for aa kunne spare insturksjoner.*/



int analyze()
{
  int n;                  /* Trenger en variable til div for lokker*/

  if(openvideoch==0)
  {
    leskonfig();          /*leser konfigfil som setter in de nodvendig
verdier inn i porgrammet*/
    openvideo();          /* Gjor klar video grabberen*/
    xstorrelse=grabsizex();
    ystorrelse=grabsizey();
```

```
for(n=0;n<21*2;n+=2)   /* prekalulerer talbell stoyfi*/
  stoyfi[n/2]=stoyfilter[n]+stoyfilter[n+1]*xstorrelse;

openvideoch=1;

/* allokerer alle buffere som trenges */

if(gjennomsnittsbildech==0)
  {
  behandlet=0;
  gjennomsnittsbilde=malloc(xstorrelse*ystorrelse*sizeof(long));
  if(gjennomsnittsbilde==0)
  {
    printf("Ikke nok minne!");
    exit(1);

  }
  differanse=malloc(xstorrelse*ystorrelse*sizeof(long));
  if(differanse==0)
  {
    printf("Ikke nok minne!");
    exit(1);

  }
  stoy=malloc(xstorrelse*ystorrelse*sizeof(long));
  if(stoy==0)
  {
    printf("Ikke nok minne!");
    exit(1);
  }

  funnet=malloc(xstorrelse*ystorrelse*sizeof(char));
  if(funnet==0)
  {
    printf("Ikke nok minne!");
    exit(1);
  }
  else for(n=0;n<xstorrelse*ystorrelse;n++)funnet[n]=0;

  winbuffer=(long *)malloc(xstorrelse*ystorrelse*sizeof(long));
  if(winbuffer==0)
  {
    printf("Ikke nok minne!");
    exit(1);
  }
  mask=malloc(xstorrelse*ystorrelse*sizeof(char));
  if(mask==0)
  {
    printf("Ikke nok minne!");
    exit(1);
  }
  else hentmask();    /* henter mask bilde hvis satt paa*/
  gjennomsnittsbildech=1;
  }


if(visvindu==1)
  {
  createRGBwin();
```

```
      visvindu=2;
      }

  }

  datafragrabber=grab(); /*Grabber bilde*/

  laggjennomsnitt();     /*Starter bilde behaldlingen*/



  if(visvindu==2)  /*Viser skjerm bilde hvis paa*/
    {
    lagbildet();
    lrectwrite(0,0, (xstorrelse)-1, ystorrelse-1, (ulong *)winbuffer);
    }
  behandlet++;


  /* Her tas avgjorelsen om programmet skal detektere noe */
  if(antallfunnet > minantallpunkter && antallfunnet <maksantallpunkter &&
behandlet>kaliberingsbilder )
    {
      behandlet=0;
      lagbildet();
      unlockbuffer();

      return 1;
    }
  unlockbuffer();
  return 0;

}

/* Avslutter programmet */
void analyseclose()
{
  closevideo();
  free(gjennomsnittsbilde);
  free(differanse);
  free(stoy);
  free(funnet);
  free(winbuffer);
  openvideoch=0;
  gjennomsnittsbildech=0;

}

void laggjennomsnitt()
{
  int x,n,y;
  long diff,st,sum;
  antallfunnet=0;

  /* kaller rutine for lyse&mokeobjekter eller bare lyse*/
  if(kunlyse==1)kunlyseobjekter();
  else lyseogmorke();
```

```
  /* Denne doble loopen oppdaterer funnet */
  /* Den setter 0xff i arry funnet der en deteksjon funnet sted*/
  /* 00 der det ikke er funnet */
  for(y=xstorrelse*2;y<((xstorrelse*ystorrelse)-
(xstorrelse*2));y+=xstorrelse)
      for(x=2;x<xstorrelse-2;x++)

      {
        sum=x+y;
        st=0;

        if(mask[sum]==0)
          {

            for(n=0;n<omraade;n++)/*finner omraade stoyen*/
            {
              st+=stoy[sum+stoyfi[n]];
            }

            /* Bestemmer om punketet blir funnet*/
        if(differanse[sum]>minimumdifferanse+(st/omraadedivisjon))
        {
          funnet[sum]=0xff;

          antallfunnet++; /* teller oppantall punketer som er detektert*/

        }
        else funnet[sum]=0;
       }

      }

}

/* Denne funksjonen oppdaterer baade stoybildet og gjennomsnittsbidet*/
void kunlyseobjekter()
{

 int n,diff;
 for(n=0;n<xstorrelse*ystorrelse;n++)

    {
      diff=(datafragrabber[n])-(gjennomsnittsbilde[n]>>16);
      gjennomsnittsbilde[n]+=diff*gjennomsnittsoppdatering;

     differanse[n]=diff;


     if(diff<0) diff=-diff;



      if (stoy[n]<(diff<<16))
      stoy[n]+=stoylaering;
      else
      if(stoy[n]>stoyglemming)stoy[n]-=stoyglemming;
    }
}
```

```
/* Denne funksjonen oppdaterer baade stoybildet og gjennomsnittsbidet*/
void lyseogmorke()
{
  int n,diff;
  for(n=0;n<xstorrelse*ystorrelse;n++)

    {
      diff=(datafragrabber[n])-(gjennomsnittsbilde[n]>>16);
      gjennomsnittsbilde[n]+=diff*gjennomsnittsoppdatering;

     if(diff<0) diff=-diff;

     differanse[n]=diff;


      if (stoy[n]<(diff<<16))
      stoy[n]+=stoylaering;
      else
      if(stoy[n]>stoyglemming)stoy[n]-=stoyglemming;
    }
}


/* intalisering av vindu */
void createRGBwin()
{
    prefsize(xstorrelse,ystorrelse);
    win = winopen("a199801");
    RGBmode();
    pixmode(PM_TTOB, 1);
    gconfig();
}

/* Denne lager bilde som skal vises paa skjerm og skrives til fil*/
void lagbildet()
{
  long teller;

      for(teller=0;teller<xstorrelse*ystorrelse;teller++)

      if(funnet[teller]>1)
      winbuffer[teller]=(datafragrabber[teller]*0x0100+0xf0)&0x00ffff;
      else
      winbuffer[teller]=datafragrabber[teller]*0x010101;

}


/* koverterer mask.TIFF og Lastere det nye bilde i inn i buffer*/
void hentmask()
{
 FILE *fileptr;
 int n,r,g,b;
 if(brukemaske==1)
   {
    system("convert -size768x576 768x576 mask.TIFF mask.RGB");
    fileptr = fopen("mask.RGB", "r");
    if(fileptr)
    {
```

```
    for(n=0;n<xstorrelse*ystorrelse;n++)
       {
       r=getc(fileptr);
       g=getc(fileptr);
       b=getc(fileptr);
       if(r==EOF||g==EOF||b==EOF)
         {
         break;
         printf("Feil paa paa fil mask.RGB\n");
         }

       if(r>b&&r>g)mask[n]=0xff;
       else mask[n]=0x00;
       }

       fclose(fileptr);
    }
    else
   {
    printf("mask.RGB, fil feil!\n");
   }
   }
   else
     {
         for(n=0;n<xstorrelse*ystorrelse;n++)
       mask[n]=0;
     }

}


/*skrive bildet som ligger i winbuffer til fil "image.RGB"*/
void skrivbildet()
{
  long teller;
  FILE *filpek;
  char *t=winbuffer;
  if (filpek = fopen("image.RGB", "w"))
    {
    for(teller=0;teller<xstorrelse*ystorrelse;teller++)
      {
      if ((fputc(t[teller*4+3], filpek)==EOF) || (fputc(t[teller*4+2],
filpek)==EOF) || (fputc(t[teller*4+1], filpek)==EOF))
      {
      printf("image.RGB, fil feil!\n");
      }
    }
    fclose(filpek);
      }
  else{
    printf("image.RGB, fil feil!\n");
  }
}


/* Skrive siste bilde fra grabber til fil "image.GRAY"*/
void lagGRAY()
{
  long x,y;
```

```
  long teller;
  FILE *filpek;
  char *t=datafragrabber;

  x= grabsizex();
  y= grabsizey();


 if (filpek = fopen("image.GRAY", "w"))
  {
    for(teller=0;teller<x*y;teller++)
      {

      if ((fputc(t[teller], filpek)==EOF) )

      {
      printf("image.RGB, fil feil\n");
      }

      }
   }
   fclose(filpek);
}



/************ KONFIG LESING ************************************/

void leskonfig()
{

    int t;
    float tall;
    FILE *fileptr;
    char data[10000];


    for(t=0;t<10000;t++)data[t]='\0';

    fileptr = fopen("a199801.conf", "r");
     if(fileptr)
   {

       t=fread(data,1,9000, fileptr);

        fclose(fileptr);

      gjennomsnittsoppdatering=(long)(0x10000*sfinn("GJENNOMSNITTSOPPDATERI
NG ",data));
      stoylaering=(long)0x10000*sfinn("STOYLAERING ",data);
      stoyglemming=(long)0x10000*sfinn("STOYGLEMMING ",data);
      omraade=(long)sfinn("OMRAADE ",data);
      if(omraade>21)omraade=21;
      omraadedivisjon=(long)0x10000*sfinn("OMRAADEDIVISJON ",data);
      minimumdifferanse=(long)sfinn("MINIMUMDIFFERANSE ",data);
      minantallpunkter=(long)sfinn("MINANTALLPUNKTER ",data);
      maksantallpunkter=(long)sfinn("MAKSANTALLPUNKTER ",data);
      brukemaske=(long)sfinn("BRUKMASKE ",data);
      kunlyse=(long)sfinn("KUNLYSE ",data);
```

```
      visvindu=(long)sfinn("VISVINDU ",data);
      if(visvindu!=0&&visvindu!=1)visvindu=0;
      kaliberingsbilder=(long)sfinn("KALIBRERINGSBILDER ",data);

   }

  else{
    printf("a199801.conf, fil feil!\n");
  }


}


float sfinn(char *test,char *data)
{
  int n,t;
  float f;

 for(n=0;n<9000;n++)
    {
      if(test[0]==data[n])
      for(t=0;t<100;t++)
        {
        if(test[t]==' ')
          {
            f=ffinn(&data[n+t]);
            return f;
          }
        if(test[t]!=data[n+t])t=1000;
        }

    }
    return 0;
}

float ffinn(char *data)
{
  float f=0;
  float f2=1;
  int n=0;
  char *t;

  while(data[n]!='\0'&&data[n]!='\n'&&!(data[n]>='0'&&data[n]<='9'))
  n++;


  while(data[n]>='0'&&data[n]<='9')
  {
    f=f*10+(data[n]-'0');
    n++;
  }
  if(data[n]=='.')
    {
    n++;

    while(data[n]>='0'&&data[n]<='9')
      {
      f2=f2/10;
```

```
      f=f+(f2*(data[n]-'0'));
      n++;
      }
    }

 return f;

}

/************************** log fil  ******************************/

FILE *logfil;

void initlog()
{
  char *c;
  time_t t;
  t=time(NULL);
  c=asctime(localtime(&t));



if ((logfil=fopen("a199801.log","a"))==NULL)
  {
    printf("a199801.log, fil feil!\n");
  }
  else
  {
  fprintf(logfil,("Startet: "));
  fprintf(logfil,c);
  fclose(logfil);
  }
}

void log(int d)
{
  int n;

  char str[100]={""};
  char *c;
  time_t t;
  t=time(NULL);
  c=asctime(localtime(&t));

  for(n=0;n<100&c[n]!='\n';n++)
    str[n]=c[n];

  str[n]='\0';

if ((logfil=fopen("a199801.log","a"))==NULL)
  {
     printf("a199801.log, fil feil!\n");
  }
  else
  {
  fprintf(logfil, "Detektnr: %i ",d);
  fprintf(logfil,str);
  fprintf(logfil," antallpunkterdetektert ");
  fprintf(logfil,"%i",antallfunnet);
```

```
  fprintf(logfil,"\n");
  fclose(logfil);

  }
}
```

main.c

Vi har en hovedfunksjon *main.c*, som et kunstig skall rundt den programbiten vi har fått i oppdrag å lage.

```
/* Main.c */
/* Dette programmet er ment aa vise hvordan det skal implementeres i Martin
THorsen program */

#include <stdlib.h>
#include <stdio.h>
#include <gl/gl.h>
#include <dmedia/vl.h>

#include "analyze.h"


/* Deklarering av variabler som programmet trenger*/

long detektnr=0;               /*Hvilket deteksjonn*/
int n;
int tall=0;
char filnavn[]={"convert -size768x576 768x576 image.RGB imagexxx.JPG\0"};
char *u=filnavn;


char *_progName;
void error_exit(void)
{
    vlPerror(_progName);
    exit(1);
}




void main(int argc, char **argv)
{
   _progName = argv[0];
  foreground();

  initlog();                     /* Legger inn tidspunkt for start*/


while(1)
  {

    n=analyze();                 /*Kjorer en analye */
    if (n==1)
      {
        /*her skal videospiller startes*/


      skrivbildet();  /*Lager et bilde image.RGB */
      log(detektnr);  /*Legger inn deteksjonen i logfila data,tid og et
dekteksjonnummer.*/



      /*Her skal bilde koverteres og sendes ut paa inter nett.*/
```

```
      /*Forelopig lagres bare programmet bildet lokalt*/
      sprintf(u, "convert -size768x576 768x576 image.RGB image%i.JPG\n",
detektnr);
      detektnr++;
      system(filnavn);
      printf("%s",filnavn); /*gir oss en medlig om deteksjon */

      }
   }


 closevideo(); /* programmet naar ikke hit med er bare med for aa demostre
at den trengs i det ferdig programmet */
}
```

grab.c
Grabbingen av bilde, og lagring av RGB-format er i *grab.c*.

```
/* Denne kode er hente fra SGI. */

#include <gl/gl.h>
#include <dmedia/vl.h>
#include "grab.h"

    static VLServer grab_svr;
    static VLPath grab_path;
    static VLNode grab_src, grab_drn;
    static VLControlValue grab_val;
    static VLBuffer grab_buffer;
    static VLInfoPtr grab_info;

    static char *grab_dataPtr;

    static int grab_c;
    static int grab_xsize;
    static int grab_ysize;


/* initaliserer videograbberen, programmet avsluttes ved feil*/

void openvideo()
{
    /* Connect to the daemon */
    if (!(grab_svr = vlOpenVideo("")))
      error_exit();

    /* Set up a drain node in memory */
   grab_drn = vlGetNode(grab_svr, VL_DRN, VL_MEM, VL_ANY);

    /* Set up a source node on any video source  */
   grab_src = vlGetNode(grab_svr, VL_SRC, VL_VIDEO, VL_ANY);

    /* Create a path using the first device that will support it */
     grab_path = vlCreatePath(grab_svr, VL_ANY, grab_src, grab_drn);

    /* Set up the hardware for and define the usage of the path */
    if ((vlSetupPaths(grab_svr, (VLPathList)&grab_path, 1, VL_SHARE,
VL_SHARE)) < 0)
      error_exit();

     grab_val.intVal = VL_PACKING_Y_8_P;
    vlSetControl(grab_svr, grab_path, grab_drn, VL_PACKING, &grab_val);

    /* Get the video size */
    vlGetControl(grab_svr, grab_path, grab_drn, VL_SIZE, &grab_val);
    grab_xsize =grab_val.xyVal.x;
    grab_ysize =grab_val.xyVal.y;

   /* Create and register a buffer for 1 frame */
   grab_buffer = vlCreateBuffer(grab_svr, grab_path, grab_drn, 1);
    if (grab_buffer == NULL)
      error_exit();
    vlRegisterBuffer(grab_svr,grab_path, grab_drn, grab_buffer);
```

```
    /* Begin the data transfer */
    if (vlBeginTransfer(grab_svr, grab_path, 0, NULL))
      {
      printf("Grabber, feil!\n");
      error_exit();
      }
}


/* Denne funksjonen grabber et bildet og returnerer en peker til dataen*/
char *grab()
{
  sginap(1);
    do {
      grab_info = vlGetNextValid(grab_svr, grab_buffer);
    } while (!grab_info);

    /* Get a pointer to the frame */
    grab_dataPtr = vlGetActiveRegion(grab_svr, grab_buffer, grab_info);




    return grab_dataPtr;

}

/*Lar grabber ta et nytt bilde */
void unlockbuffer()
{
  /* Finished with frame, unlock the buffer */
    vlPutFree(grab_svr,grab_buffer);

}




/* Frigjor grabberen igjen */
void closevideo()
{
 vlEndTransfer(grab_svr, grab_path);
     /* Cleanup before exiting */
    vlDeregisterBuffer(grab_svr,grab_path, grab_drn, grab_buffer);
    vlDestroyBuffer(grab_svr, grab_buffer);
    vlDestroyPath(grab_svr, grab_path);
    vlCloseVideo(grab_svr);
}

/* to funksjonener som returnerer opplosningen*/
int grabsizex()
{
  return grab_xsize;
}


int grabsizey()
{
  return grab_ysize;
```

```
}
```

(Vedlegg)

Brukermanual

for programmet "ana",

Prosjektgruppe a199801

Objektdetektering

Hurtigstart, gjennomgående forklaringer og appendix med dypere forklaringer

Innholdsfortegnelse for brukermanual

Hurtigstart - 3 steg til start av programmet

Steg 1 - Sjekk konfig-filen

Stå i riktig katalog (ana) og editer konfig-filen:

/disk6/usr/people/a199801/analyse/a199801.conf

Sørg for at følgende er stilt inn:

```
BRUKMASKE  0
VISVINDU  1
```

Husk å lagre

Steg 2 - Sjekk kameraet

Sørg for at kameraet er koblet til, og har strøm.

Steg 3 - kjør programmet

Stå i analyse-katalogen (`/disk6/usr/people/a199801/analyse/`) og skriv "`ana`". Programmet vil da starte, og et vindu på skjermen viser hva kameraet filmer. Dersom noe ikke virker som det skal, se seksjonen "Feilsøking".

Oppkobling av utstyret

Generelt om nettverket
Begge datamaskinene er koblet sammen i et lite lokal nettverk (ethernet). Dette nettverket er koblet til en ruter, som også har en isdn bri (basic rate interface) tilkobling. Dette interfacet er tilknyttet en vanlig dobbel isdn-linje (2b- og en d-kanal).

Pentium 100
Denne maskinen skal være tilknyttet alle de andre sensorene som skal være på målestasjonen. Dette skjer via et io-kort som sitter i PC'en.

CCD-kameraet

Tilknytningen til videokameraet

Fra linsen går det en ledning som skal tilkobles kameraet. Denne ledningen kontrollerer blenderen i linsa, (autoiris). Fra CCD-kameraet går det en 75 ohms coax-kabel til videospillerens euroscart 2. Fra videospillerens euroscart 1 går det en phonoledning til Indymaskinens framegrabber.

Innstilling av kameraet

5-bit-switch (Sett ovenfra)
AGC - PÅ
INT/LL - PÅ (INT)
ELC/ALC - PÅ (ALC)
VIDEO/DC - AV (DC)
Hi-Z/75ohm - PÅ (75 ohm)

Dreiepotmetre (Sett ovenfra)
V.PHASE - Cirka kl. 11
ALC LEVEL - Cirka kl. 11

Lokalisering av filene

Filene som hører til vårt prosjekt ligger på hessdalen.hiof.no (ip nr. 158.36.16.146) under "/disk6/usr/people/a199801/analyse". Man kan bruke ftp for å hente filene fra hessdalen.hiof.no, dersom man har rettighetene. Det er også dette stedet bildene og loggfilen blir lagret.

Beskrivelse av filene

Programmet innholder disse filene: main.c, analyze.c, analysz.h, grab.c, grab.h

Main.c

Dette er fila som styrer hovedprogrammet. Her er programmet skrevet slik at det skal være lett å se hvordan det skal implementeres i et annet program. De funksjonenen som som skal kalles, heter: int analyse();, closevideo();, initlog();, log(int d);, skrivbilde();

Det skal ikke vært nødvendig å studere andre deler av programmet.

Analyze.c

Denne filen inneholder bildebehandlingdelen, konfig-innlesingen og loggfilsystemet. Den bruker også funksjonene i filen grab.c.

int analyse();

Denne funksjonen returnerer 1 når programmet har detektert noe, og 0 når den ikke har detektert noe. Første gangen den blir kalt vil programmet automatisk sjekke om video-grabberen er i bruk, og allokere nødvendig mengde minne. Dersom noe av dette går feil, vil programmet stoppe.

closevideo()

Her blir minnet og grabberen frigjort.

initlog()

Denne åpner fil a199801.log, og legger dette til på slutten av filen: "Startet: Thu Jun 4 12:50:58 1998". (Klokken og datoen på det tidspunktet programmet ble startet).

Loggsystemet er laget slik at logg-filen ikke sletter seg hver gang en starter programmet på ny. Den nye loggen fortsetter bare på den gamle.

log(int d)

Denne legger inn et id-nummer i loggfilen , antall punkter som er detektert og tid og dato.
"D" er da id-nummeret.

void skrivbilde()

Denne funksjonen brukes til å lage en fil som heter image.RGB. Denne filen inneholder bildet som programmet har trigget på. De punktene som programmet har detektert på, har fått endret farge til rødt eller gult. Oppløsningen er 768x576 piksler. Data-formatet er konvertert fra RAW til RGB. Dette vil si at dataene ligger upakket og hver piksel har fått tre bytes, (1 byte med rød, 1 byte med grønn og en med blå), og uten informasjon om oppløsnig. For å konvertere dette til et annet format, kan man bruke "`convert`" som Martin R. Thorsen har installert på Indy-maskinen.

Eksempel på konvertering fra RGB til JPEG:

```
convert -size768x576 768x576 image.RGB image.JPG
```

Eksempel på konvertering fra RGB til GIF:

```
convert -size768x576 768x576 image.RGB image.GIF
```

void lagGRAY()

Denne funksjonen grabber et bilde og lagrer som `image.GRAY`. Data-formatet er `Raw.GRAY`. Dette vil si at hver pixel har fått en byte og det er kun gråtoner som er lagret, uten informasjon om oppløsning.

Eksempel for å konvertere dette til JPEG:

```
convert -size768x576 768x576 image.GRAY image.JPG
```

Eksempel for å konvertere dette til GIF:

```
convert -size768x576 768x576 image.GRAY image.GIF
```

Grab.c

Denne inneholder et program som er modifisert av oss. Det er laget av SGI. Det var et program som tok et enkelt bilde . Vi forandret det til svart/hvitt.

Headerfilene (*.h)

Disse filene inneholder funksjonsdeklarasjonene i kildekodefilene (*.c). Normalt trenger man ikke å endre på disse.

Testing av utstyret

Når utstyret er koblet opp riktig, kan man teste det for å sjekke at det fungerer som det skal. Det første man da må gjøre, er å sjekke at konfig-filen er grovt sett riktig innstilt.

Grovinnstilling av konfig-filen (a199801.conf)

Det er ikke noen fasit på hvordan man kan sette opp verdiene i konfig-filen, man må bruke skjønn. Her følger en beskrivelse på hvordan man grovt sett innstiller programmet. Nærmere forklaringer av de forskjellige innstillingene finnes under appendix. Når man skal endre på konfig-filen, gjør man dette med en editor, f.eks. "`emacs a199801.conf`".

Dersom man er usikker på hvordan man skal starte opp programmet, så se på seksjonen "Oppstart av systemet"

1. Sett verdiene til følgende :

```
GJENNOMSNITTSOPPDATERING 0.10
OMRAADE  13
OMRAADEDIVISJON 2
MINIMUMDIFFERANSE 10
STOYLAERING  2.0
STOYGLEMMING  0.15
MINANTALLPUNKTER 10
MAKSANTALLPUNKTER 10000
BRUKMASKE  0
KUNLYSE  0
KALIBRERINGSBILDER 15
VISVINDU  1
```

2. Starte programmet

Kjør programmet `ana`. Vent 5 minutter, da programmet trenger tid til å kalibrere seg.
På bildet kan man da se om det blir detektert altfor mange punkter.

3. Reduser hva støyen har å si for deteksjonene

Øk `OMRAADEDIVISJON` med 30 til 100 %.

4. Starte programmet igjen

Kjør programmet vent 5 min. Nå vil antagelig programmet detektere færre uønskede punkter.

5. Gjenta prosessen over

Forsett med punkt 3 og 4, helt til en ser at programmet ikke detekterer uønskede punkter. Hvis det er noe område i bildet som det ikke er mulig å fjerne på denne måten, kan man bruke et maskebilde for å fjerne de uønskede punktene. (Se "Bruk maske").

6. Støylæring og støyglemming

Nå kan man begynne å stille på `STØYLAERING` og `STØYGLEMMING`. Her må en bruke skjønn og prøve seg frem. Desto høyere verdi på `STØYLAERING`, desto raskere blir forandringer i bildet tolket som støy. Jo mindre `STØYGLEMMING`, jo lengre vil programmet huske at det er støy i et område. Kameraets vinkel mot himmelen har mye å si, da dette bestemmer skyenes pixelhastighet i bildet. Disse innstillingene bør en ikke røre før punkt 5 er fullbyrdet.

Andre innstiller, som `MINANTALLPUNKTER` som programmet skal reagere på, setter man etter eget (fornuftige) ønske, f. eks. 15.

Maskebildet - grabbing, henting, manipulering og tilbakesending

Maskebildet bruker man for å kunne "tegne vekk" deler av bildet, slik at systemet ikke skal trigge på endringer i disse delene. Eksempler på ting man kan, (og bør) maskere vekk, er veier og husvinduer. Under følger en forklaring på hvordan man kan få tak i et relevant, maskbart bilde, og prosessen rundt det.

Grabbing

For å få et maskebilde som er relevant for systemet, kan man ikke grabbe et bilde før hele målestasjonen er satt opp, og alt er klart. For å grabbe et bilde, skriver man bare "`grabmask`" fra analyse-katalogen. Dette genererer to bilder på harddisken, `mask.TIFF` og `mask.RGB`. (Det siste bildet er bare for konverteringens skyld).

Henting

Man henter enklest bildet som skal manipuleres, `mask.TIFF`, ved å bruke et ftp-program. Grunnen til at man henter bildet til en annen maskin, er at Indy-maskinen ikke har noen gode bildemanipuleringsprogrammer. Med ftp-programmet går man til analyse-katalogen og laster ned "`mask.TIFF`". Ftp-programmer finner man gratis på Internettet, ved å søke etter f.eks. "Cuteftp download official" på en websøker.

Manipulering

For å tegne inn de delene av bildet som skal ekskluderes fra deteksjonsrutina, trenger man bare å tegne over disse delene med en rødfarge. Man må tegne over HELE området som skal bort. "En rødfarge" vil si at den røde delen i RGB-verdien er større enn den blå og grønne. (F.eks. *240*, *154*, *36*). Man må også konvertere bildet til 24-bit. Man kan bruke et hvilket som helst program for å gjøre dette, men vi anbefaler "Paint Shop Pro". Dette er et meget enkelt og effektivt program som man kan laste ned en prøveversjon fra Internettet ved å søke etter f.eks. "Paintshop pro download official".

Tilbakesending

Ved å bruke et ftp-program sender man tilbake bildet `mask.TIFF` (case-sensitiv) til analyse-katalogen. Dermed er man ferdig. Når så analyseprogrammet starter, (ved å skrive "`ana`"), og konfig-filen er innstilt til å bruke maskebilde, (`BRUKMASKE 1`), vil programmet automatisk benytte seg av maskebildet `mask.TIFF` som ligger i katalogen.

Oppstart av programmet

Dersom programmet ikke er kompilert
Da må det kompileres, og det gjør man ved å skrive "`make`" i analyse-katalogen. `make` er en kommando som blant annet henter informasjon fra en "`Makefile`". I denne filen står det hvilke kildefiler som skal kompileres, og hvilke "includes" som skal være med på prosjektet som skal kompileres. Denne filen er det ikke nødvendig å endre på, det holder altså å skrive "`make`". Noen få ganger hender det at man må slette de gamle "`*.o`"-filene og den gamle `ana`, (den kjørbare filen).

Dersom programmet er kompilert
Da trenger man ikke å skrive annet enn "`ana`" i analyse-katalogen. Programmet vil da starte opp, og man vil se et vindu på skjermen med bildet som kameraet sender, dersom man har valgt dette i konfig-filen. Dette bildet vil oppdatere seg cirka en gang annet hvert sekund.

Kontinuerlig drift av systemet

Programmets virkemåte

Dersom noe detekteres (når programmet kjører), vil bildet på skjermen vise dette området ved å markere det med rødt. Slik kan man se at programmet faktisk har en deteksjon. Både programmets oppstartstid, deteksjonens id-nr. og antallet pixler den har trigget på noteres i loggfilen (`a199801.log`). Programmet vil også lagre en kopi av bildet i jpeg-format, slik at man etterpå kan se på det i et bildefremvisningsprogram, (vi brukte faktisk mest Netscape, da dette var enklest). Denne funksjonen som lagrer bildet på harddisken skal egentlig gå under `main.c`, som, når alt er ferdig, ikke skal være definert av oss. Dermed er det ikke sikkert at den fungerer på samme måte når systemet er i operasjon. Nærmere opplysninger om hvordan det da fungerer, må man finne under dokumentasjonen av `main.c`.

Viktige ting under lengre tids drift

Programmet

Det er ikke mange ting man trenger å gjøre med programmet dersom det skal kjøre over lenger tid av gangen. Under forutsetning at man har klart å stille inn konfig-filen noenlunde riktig, så vil man ikke få så mange deteksjoner. Det eneste man da må holde øye med, er at Indy-maskinen ikke går tom for diskplass. Da hvert bilde tar cirka 30 kilobyte, er det på det nåværende tidspunkt plass til omtrent 8000 bilder. Disse bildene kan slettes via ftp til hessdalen.hiof.no, så man trenger ikke å være på stedet fysisk.

Kameraet

Det eneste man må sørge for i forbindelse med kameraet, er at det har fri sikt. Dette innebærer at man må antakelig pusse glassåpningen i kamerahuset av og til, og fjerne eventuelle dyr, fugler eller insekter som måtte være tiltrukket av det oppvarmede kamerahuset.

Få, eller ingen deteksjoner over lenger tid

Dersom det er ingen, eller bare få deteksjoner over et lenger tidsrom, må man spørre seg om noe er galt. Det kan være at harddisken er full, og da stopper mest sansynligvis programmet, eller kameraet fungerer ikke lenger. Det kan også være naturlige forhold, slik som endring av årstider og lysforhold, som gjør at nåværende innstillinger i konfig-filen ikke fungerer lenger. For andre feil, se "Feilsøking".

Feilsøking

Vi forutsetter at programkoden som er innbefattet med vårt prosjekt fungerer som den skal, men dersom den må endres, så beskrives hvordan man gjør dette under "Modifisering av kildekoden".

Vi velger med å starte med å beskrive den mest sannsynlige feilen og mulige årsaker.

Programmet starter ikke:

- Det er ikke nok ledig minne.
- Programmet får ikke tilgang til videograbberen. (Et annet program bruker den?)
- Ikke nok diskplass på Indymaskinen. (Dette stopper antakelig resten av maskinen også.)
- Flere programmer prøver å allokere grabberen til seg samtidig. (Dette er en svakhet ved operativsystemet.)

Programmet starter, men vinduet på monitoren er svart:

- Sjekk at kameraet har strøm.
- Sjekk at coax-kabelen er tilkoblet, og ikke brutt.
- Sjekk at kablene er riktig tilkoblet videospilleren
- Sjekk at kabelen mellom videospilleren og Indymaskinen er riktig koblet.
- Sjekk at grabberens videokilde stemmer overens med inngangen på Indymaskinen, (se "Velg riktig videokilde").
- Sjekk at alle verdiene i konfig-filen er noenlunde fornuftig.
- Sjekk at det er fri sikt foran kameraet.
- Sjekk at kameraet faktisk er i funksjonsmessig stand, (ved f.eks. å tilkoble en tv direkte.)

Programmet starter og går, men det detekterer ikke riktig:

- Sjekk at riktig videokilde på grabberen er valgt, (se "Velg riktig videokilde").
- Sjekk innstillingene i konfig-filen.
- Sjekk at konfig-filen eksisterer.
- Sjekk at videosignal-kabelen til kameraet er uten brudd, og at den er av riktig type (75 ohm coax).
- Dersom man har valgt å bruke maskebilde, og maskebildet mangler fra katalogen.
- Dersom man har valgt å bruke maskebilde, og det er feil i dette bildet, (feil område skravert, eller at man ikke har brukt en rødfarge for skraveringen).
- Kameraet står for ustødig.
- For mange prosesser går samtidig på Indymaskinen, slik at tidsforskjellene blir for store i analyseprogrammet.
- Systemet har stått så lenge, at kameraet ikke lenger har samme bilderamme som maskebildet.

Velg riktig videokilde

SGI indy 100 har to innganger for video. En inngang er beregnet for ekstern video, og en er beregnet for det lille kameraet (indy-cam) som følger med maskinen. Dersom maskinen er innstilt på å ta bilder fra kameraet, vil programmet vårt også hente bilder fra denne kilden. For å endre dette må vi ha startet programmet "`capture`", (vanligvis et ikon oppe til høyre i desktopen). Trykk på "Actions/Settings". Da kommer det opp et nytt vindu som heter "Capture Movie Settings", trykk så på "video." Et nytt vindu kommer opp - "Video Panel". Velg "VINO DEVICE CONTROLS" til "Analog Source", "Video Format" til "Composite", og "Input Timing" til "PAL (625)".

Modifisering av kildekoder

Man kan endre på kildekoden til programmet, dog er dette noe vi ikke uten videre anbefaler. Man har veldig mange innstillinger og kombinasjoner av disse å prøve i konfig-filen, før man bør tenke på å endre kildekodene. Men dersom man virkelig vil endre på disse, kan man gjøre dette enklest ved å åpne koden i en editor, f.eks. `emacs`. Når man er ferdig med endringene må man kompilere programmet før man kjører det, (se "Oppstart av programmet").

APPENDIX A

Dypere forklaringer til konfigurering av programmet

Konfigurasjons-filen heter "`a199801.conf`", og ligger i sammer katalog som bildeanalyse programmet, ("`/disk6/usr/people/a199801/analyse/`").

Oversikt over innstillingene og mulige valg

Her følger en oversikt over innstillingene og de valgmulighetene man har på hver:

```
GJENNOMSNITTSOPPDATERING  <  0 , 1 >
OMRAADE                   1, 5, 9, 13, 21
OMRAADEDIVISJON           > 0
MINIMUNDIFFERANSE         > 0
STOYLAERING               > 0
STOYGLEMMING              > 0
MINANTALLPUNKTER          > 0
MAKSANTALLPUNKTER         > 0
BRUKMASKE                 0 , 1
KUNLYSE                   0 , 1
VISVINDU                  0 , 1
KALIBERINGSBILDER         > 1
```

GJENNOMSNITTSOPPDATERING

Denne kan stilles fra 0 til 1. Denne faktoren oppdaterer gjennomsnittsbildet.

endring = nåværende bilde - gjennomsnittsbildet

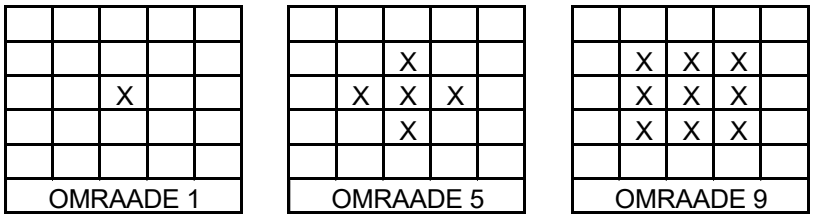
gjennomsnittsbilde = gjennomsnittsbilde + (endring `GJENNOMSNITTSOPPDATERING`)*

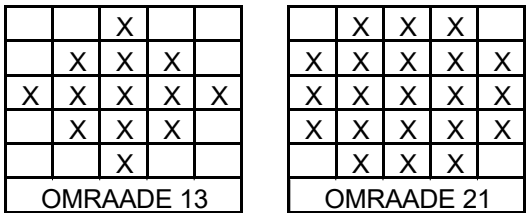
Dersom faktor en er "`1`" vil gjennomsnittsbildet alltid bli det samme som bildet som akkurat er tatt av grabberen, og hvis faktoren er "`0`" vil gjennomsnittsbildet ikke forandre seg . Det er ikke noen fasit på hvilken verdi som er best, men vi har funnet ut at denne faktoren bør ligge rundt `0.05` til `0.15`. Kameraets motiv har innvirkning på denne faktoren. Når faktoren er liten, vil programmet kunne detektere objekter som beveger seg saktere men blir mer følsom ovenfor støy, slik som vær-fenomener.

OMRAADE

Programmet tar hensyn til støy. Hvert punkt har en verdi som beskriver støyverdien i punktet. `OMRAADE` bestemmer hvor stort område rundt hvert punkt som skal tas hensyn til. `OMRAADE` kan settes til følgende verdier: `1`, `5`, `9`, `13`, `21`. Dersom verdien er satt til `1`, vil støy bli hentet

ut fra punktet selv. Hvis verdien er satt til `13`, vil programmet detektere støy fra 12 punkter rundt selve punktet. Ved å øke området, vil programmet kunne lettere kategorisere ting som sky, sol, måne, regn og snø som støy, og la være å trigge (reagere) på det.





Midtpunktet i hver tabell er selve punktet man tester på om det er noe til stedet.

Programmet vil hente ut støy fra de punktene som er merket med X og summere disse. Vi har kalt dette områdestøy. Anbefalt verdi er `13`.

OMRAADEDIVISJON

`OMRAADEDIVISJON` er den verdien som bestemmer nevneren i en divisjon med summen av støyen i et område som teller. Dersom en øker `OMRAADE` må en også øke `OMRAADEDIVISJON`. Øker en `OMRAADE` fra `5` til `9`, må en øke `OMRAADEDIVISJON` med nesten det dobbelte. Hvis programmet detekterer for lite, kan man forsøke å øke `OMRAADEDIVISJON`'en, og hvis det detekterer for mange ting, kan den minskes.

MINIMUNSDIFFERANSE

`MINIMUMSDIFFERANSE`'n brukes til å hjelpe til å justere hvordan programmet skal detektere et punkt. Detekteringen skjer slik:

$|endring| < (MINIMUMSDIFFERANSE + (områdestøy / OMRAADEDIFFERANSE))$

`MINUMSDIFFERANSE` bestemmer hvor stor endring i bildet det må være før det skjer noen endring.

STOYLAERING, STOYGLEMMING

Dette er faktorer som bestemmer hvordan støybildet blir oppdatert. Dersom endringen i punktet man sjekker er større enn støyen i punktet, så øk støyen i punktet med

STOYLAERING, og hvis endring er mindre enn støyen så mink støyen i punktet med STOYGLEMMING.

STOYLAERING bør være større enn STOYGLEMMING. Dette er fordi at når programmet har økt støybildet i noe punkter, er det ønskelig at de skal vare en stund slik at vi ikke får detektert uønskede objekt. Ved å sette STOYLAERING lavere, vil det ta lengre tid før støyen i et punkt kan øke, og kalibreringstiden vil øke. Ved å sette STOYLAERING høyere, vil det ta kortere tid før støyen i et punkt kan øke, og kalibreringstiden vil minke. Det samme skjer med STOYGLEMMING. Anbefalte verdier for STOYLAERING er fra 0.002 til 4, og at man setter STOYGLEMMING til cirka STOYLAERING / 10.

MINANTALLPUNKTER, MAKSANTALLPUNKTER

For at programmet skal detektere et objekt må antall punkter detektert være større enn MINANTALLPUNKTER og mindre enn MAKSANTALLPUNKTER.

BRUKMASKE

Denne har to innstillinger: 0 = av og 1 = på. Dersom den er på laster, programmet inn en fil som heter mask.TIFF. TIFF er formatet på bildet. Dette bilde må være 768x576 pixler, men det kan være både 16.2Millioner- eller 256-fargers format. Dette bildet avgjør om hvor det skal detekteres. Hvis fargen i et punkt på dette bildet har mere rødt i seg enn blått eller grønt, vil programmet ikke gi deteksjon på dette punktet. Hvis mask.TIFF inneholder mange skraverte områder, vil programmet bruke mindre prosessorkraft. Det vil da gå raskere, som kan medføre at andre innstillinger må endres litt.

KUNLYSE

Denne har også bare to innstillinger: 0 = av og 1 = på. Når den er "på", kan programmet kun detektere punkter som er lysere enn bakgrunnen. Hvis endringen er negativ, så vil ikke programmet detektere punktet. Denne funksjonen er laget for å ta bort de fleste deteksjoner av fugler på dagen. Er den "av", så detekterer den både mørkere og lysere punkter. Hvis en stiller om fra "av" til "på", så bør man justere på følsomheten til de andre innstillingene. Eksempel: OMRAADEDIVISJON opp, og/eller MINIMUNDIFFERANSE ned.

VISVINDU

Denne har to innstillinger: 0 = av og 1 = på. Dersom den står "på", blir det vist et vindu på skjermen som viser det bildet som sist ble behandlet i svart/hvitt. Hvis det var noen punkter som ble detektert på dette bildet, vil de bli farget med en farge fra rødt til gult. De detekterte punktene mister blå farge og får satt rød fargen til max.

KALIBRERINGSBILDER

Dette er antall bilder som må bli behandlet av programmet før det får lov til å detektere noen objekter. Når et objekt er blitt detektert, vil det også ta en tilsvarende mengde bilder før programmet kan detektere et nytt objekt.

Tidsbruk i programmet

Programmet er laget slik at uansett hvordan bildedataen er, vil programmet bruke like lang tid på hvert bilde, det vil si like mye prosessorkraft på vært bilde. Dette er gjort bevisst for å få likt tidsrom mellom bildene, slik at detekteringen skal gå lettere, med hensyn til innstillinger i konfig-filen.

APPENDIX B

Eksempel på logg-fil (a199801.log)

```
Startet: Thu Jun  4 16:08:33 1998
Detektnr: 0 Thu Jun  4 16:09:03 1998 antallpunkterdetektert 236
Detektnr: 1 Thu Jun  4 16:09:37 1998 antallpunkterdetektert 14
Detektnr: 2 Thu Jun  4 16:11:13 1998 antallpunkterdetektert 56
Detektnr: 3 Thu Jun  4 16:11:53 1998 antallpunkterdetektert 29
Detektnr: 4 Thu Jun  4 16:13:19 1998 antallpunkterdetektert 61
Detektnr: 5 Thu Jun  4 16:14:07 1998 antallpunkterdetektert 31
Detektnr: 6 Thu Jun  4 16:14:55 1998 antallpunkterdetektert 26
Detektnr: 7 Thu Jun  4 16:15:36 1998 antallpunkterdetektert 32
Detektnr: 8 Thu Jun  4 16:16:07 1998 antallpunkterdetektert 116
Detektnr: 9 Thu Jun  4 16:16:39 1998 antallpunkterdetektert 50
Detektnr: 10 Thu Jun  4 16:17:11 1998 antallpunkterdetektert 49
Detektnr: 11 Thu Jun  4 16:17:46 1998 antallpunkterdetektert 16
Detektnr: 12 Thu Jun  4 16:18:18 1998 antallpunkterdetektert 22
Detektnr: 13 Thu Jun  4 16:19:09 1998 antallpunkterdetektert 13
Detektnr: 14 Thu Jun  4 16:19:55 1998 antallpunkterdetektert 27
Detektnr: 15 Thu Jun  4 16:21:07 1998 antallpunkterdetektert 13
Detektnr: 16 Thu Jun  4 16:21:47 1998 antallpunkterdetektert 14
Detektnr: 17 Thu Jun  4 16:22:53 1998 antallpunkterdetektert 13
Detektnr: 18 Thu Jun  4 16:24:03 1998 antallpunkterdetektert 12
Detektnr: 19 Thu Jun  4 16:24:59 1998 antallpunkterdetektert 11
Detektnr: 20 Thu Jun  4 16:25:38 1998 antallpunkterdetektert 21
Detektnr: 21 Thu Jun  4 16:26:15 1998 antallpunkterdetektert 22
Detektnr: 22 Thu Jun  4 16:27:07 1998 antallpunkterdetektert 21
Detektnr: 23 Thu Jun  4 16:27:39 1998 antallpunkterdetektert 36
Detektnr: 24 Thu Jun  4 16:28:10 1998 antallpunkterdetektert 22
Detektnr: 25 Thu Jun  4 16:28:42 1998 antallpunkterdetektert 38
Detektnr: 26 Thu Jun  4 16:29:13 1998 antallpunkterdetektert 29
Detektnr: 27 Thu Jun  4 16:30:57 1998 antallpunkterdetektert 14
Detektnr: 28 Thu Jun  4 16:31:28 1998 antallpunkterdetektert 26
Detektnr: 29 Thu Jun  4 16:32:00 1998 antallpunkterdetektert 79
Detektnr: 30 Thu Jun  4 16:32:32 1998 antallpunkterdetektert 57
Detektnr: 31 Thu Jun  4 16:33:03 1998 antallpunkterdetektert 58
Detektnr: 32 Thu Jun  4 16:33:34 1998 antallpunkterdetektert 147
Detektnr: 33 Thu Jun  4 16:34:06 1998 antallpunkterdetektert 101
Detektnr: 34 Thu Jun  4 16:34:37 1998 antallpunkterdetektert 63
Detektnr: 35 Thu Jun  4 16:35:09 1998 antallpunkterdetektert 29
Detektnr: 36 Thu Jun  4 16:36:01 1998 antallpunkterdetektert 11
Detektnr: 37 Thu Jun  4 16:36:43 1998 antallpunkterdetektert 12
Detektnr: 38 Thu Jun  4 16:37:22 1998 antallpunkterdetektert 12
Detektnr: 39 Thu Jun  4 16:38:44 1998 antallpunkterdetektert 13
Detektnr: 40 Thu Jun  4 16:39:28 1998 antallpunkterdetektert 14
Detektnr: 41 Thu Jun  4 16:39:59 1998 antallpunkterdetektert 55
Detektnr: 42 Thu Jun  4 16:40:31 1998 antallpunkterdetektert 123
Detektnr: 43 Thu Jun  4 16:41:02 1998 antallpunkterdetektert 43
Detektnr: 44 Thu Jun  4 16:41:39 1998 antallpunkterdetektert 16
Detektnr: 45 Thu Jun  4 16:42:10 1998 antallpunkterdetektert 37
Detektnr: 46 Thu Jun  4 16:42:42 1998 antallpunkterdetektert 67
Detektnr: 47 Thu Jun  4 16:43:13 1998 antallpunkterdetektert 42
Detektnr: 48 Thu Jun  4 16:44:02 1998 antallpunkterdetektert 21
```

APPENDIX C

Eksempel på konfig-fil (a199801.conf)

```
GJENNOMSNITTSOPPDATERING 0.12
OMRAADE  13
OMRAADEDIVISJON 6
MINIMUMDIFFERANSE 10
STOYLAERING  2.0
STOYGLEMMING  0.15
MINANTALLPUNKTER 10
MAKSANTALLPUNKTER 10000
BRUKMASKE  0
KUNLYSE  0
KALIBRERINGSBILDER 15
VISVINDU  1
```

APPENDIX D

Eksempel på maskebilde (mask.TIFF)

Her har vi valgt å maskere bort et av de tre trærne i forkant i bildet, da disse støyer mye når de blåser frem og tilbake. I en relevant situasjon ville vi nok ikke gjort dette, men heller endret konfig-filen.

APPENDIX E

Eksempel på deteksjon (bilde)

Her ser vi at programmet har trigget på noen som er på vei inn hovedinngangen på skolen. Flekken blir av en farge fra gul til rød.