Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Informasjonsteknologi - Signalbehandling og automatisering	Vårsemesteret, 2015 Åpen
Forfatter: Ragnhild Bechmann	…… (signatur forfatter)
Fagansvarlig: Ivar Austvoll Veileder(e): Ivar Austvoll	
Tittel på masteroppgaven: Objektgjenkjenning av sykler ved anvendelse av egenskapsdetektorene BRISK, SUSAN og SURF Engelsk tittel: Object recognition of bicycles using the feature detectors BRISK, SUSAN and SURF	
Studiepoeng: 30	
Emneord: - Egenskapsdetektorer - Objektgjenkjenning - Bildebehandling - Segmentering - Bakgrunnssubtraksjon - BRISK, SUSAN og SURF	Sidetall: 101 + vedlegg/annet: 21 + 7zip + minnepenn (originale bilder) Stavanger, 15. Juni 2015

Objektgjenkjenning av sykler ved anvendelse av egenskapsdetektorene BRISK, SUSAN og SURF

Masteroppgave vår 2015

Ragnhild Bechmann

15. juni 2015

Universitetet i Stavanger

Sammendrag

Egenskapsdetektorer er detektorer som finner fremtredende punkter med en type egenskap i bilder. Disse egenskapspunktene kan brukes til mange applikasjoner innen maskinsyn. I denne oppgaven er det valgt å bruke egenskapspunkter til objektgjenkjenning. Objektet som skal gjenkjennes er en sykkel. Målet er at en skal kunne bruke dette til å telle syklister på sykkelvei.

For å finne objektet testes tre egenskapsdetektorer opp mot hverandre: BRISK, SUSAN og SURF. Dette er detektorer som alle er skala– og rotasjonsinvariante. I tillegg er metodene bak detektorene svært ulike. Disse testes opp mot lysendring, okklusjon og ved figurer som ligner sykkelhjulet. For å detektere hjulene ut i fra egenskapspunktene brukes Hough-transformens sirkeldeteksjon og for å segmentere sykkelen brukes bakgrunnssubtraksjon.

SUSAN-detektoren blir valgt på bakgrunn av at den finner flest punkter ved sykkelhjulene og er mest stabil i forhold til sirkeldeteksjonen. Detektoren finner mange punkter utenfor objektet i tillegg. Derfor optimaliseres detektoren i forhold til geometrisk terskel, gråverditerskel samt maskestørrelse. I tillegg legges det til kriterier til sirkeldeteksjonen. Sykkeldeteksjonen fungerer godt med SUSAN-detektoren etter optimalisering. I tillegg blir utklipp av interessant område i bildet testet. Dette gir det beste resultatet.
Metoden er testet på bilder tatt ved en sykkelvei i Stavanger og alle syklene blir detektert.

Forord

Med denne oppgaven fullfører jeg mitt masterstudiet i informasjonsteknologi, signalbehandling og automatisering, ved Universitetet i Stavanger, våren 2015.

Jeg vil takke veilederen min, Ivar Austvoll, for god veiledning, råd og tilbakemelding gjennom arbeidet mitt på masteroppgaven.

Jeg vil også takke kjæresten for råd og støtte gjennom oppgaven og medstudenter for at de har holdt motivasjonen min oppe.

Stavanger, 15 Juni 2015

Ragnhild Bechmann

Innhold

Figurer

Ord og forkortelser

'SIFT' - Scale-Invariant Feature Transform

'SURF' - Speeded Up Robust Features

'GLOH' - Gradient Location and Orientation Histogram

'SVM' - Support Vector Network

'AGAST' - Adaptive and Generic Corner Detection Based on the Accelerated Segment Test

'FAST' - Features from Accelerated Segment Test

'BRIEF' - Binary Robust Independent Elementary Features

'BRISK' - Binary Robust Invariant Scalable Keypoints

'SUSAN' - Smallest Univalue Segment Assimilating Nucleus

'SLAM' - Simultaneous Localization and Mapping

Kapittel 1

Innledning

Egenskapsdetektorer er detektorer som finner fremtredende punkter med en type egenskap i bilder. Egenskapene er for eksempel hjørner eller kanter. Disse egenskapspunktene kan brukes til mange applikasjoner innen maskinsyn. I denne oppgaven er det valgt å bruke egenskapspunktene til objektgjenkjenning. Objektet som skal bli gjenkjent er en sykkel. Grunnen til at objektet sykkel er valgt er tanken på at egenskapsdetektorer kan anvendes ved telling av syklister i trafikken. Statens vegvesen teller syklistene for å kartlegge utviklingen av andel syklister i trafikkbildet. I det kartleggingsarbeidet brukes induktive sensorer. Svakheter ved disse sensorene er at det visse sykler som ikke registreres (for eksempel karbonsykler) og høyspentkabler i bakken kan forstyrre tellingene. En av svakhetene er og at det er vanskelig å telle dersom flere syklister passerer over sensoren samtidig. Dermed er det interessant å teste om egenskapspunkter gjennom bilder kan brukes i denne sammenheng. I denne oppgaven vil jeg ta for meg tre ulike egenskapsdetektorer og teste disse opp mot gjenkjenning av en sykkel. Delen av objektet som skal detekteres er hjulene.

1.1 Generelt om egenskapsdeteksjon

Det å kunne finne samsvarende punkter i to bilder tatt av samme scene eller samme objekt er del av mange applikasjoner innen maskin syn. Disse applikasjonene kan være kamerakalibrering, 3D rekonstruksjon, bevegelsesporing eller objektgjenkjenning. Applikasjonene er avhengige av at de riktige bilderegionene blir plukket ut og at detektoren som finner punktene er både raske, repeterbare og robuste mot bildetransformasjoner. Bildetransformasjoner som detektoren bør være robust mot er rotasjon, skalaendring, lysendring, komprimering og okklusjon.

Punktene som blir funnet i de ulike bilderegionene blir kalt egenskapspunkter. Dette er fremtredende punkter i bildet som inneholder spesielle egenskaper. Egenskapspunktene vil ofte være isolerte punkter, kontinuerlige kurver eller sammenkoblede bilderegioner. Noen eksempler på egenskaper er kanter, hjørner, 'blobs' (en region av et digitalt bilde der noen egenskaper er konstant eller varierer innenfor et forhåndsbestemt område av verdier[3]) eller 'ridges' (rygg) [4].

Søkingen etter diskrete bildepunktskorrespondenser (egenskapspunkter) kan bli delt inn i tre hovedtrinn. Ved første trinn finner en karakteristiske punkter i bildet, dette er det detektoren som gjør. Deretter blir området omkring disse punktene representert med en egenskapsvektor. Denne vektoren kalles en deskriptor og det er denne som beskriver egenskapspunktet. Ved å ha denne beskrivende vektoren av punktet vil detektoren kunne finne det samme punktet omigjen. Det er viktig at den er karakteristisk samtidig som den er robust mot støy og deformasjon av bildet. I det tredje trinnet vil vektorene bli matchet mellom ulike bilder [5]. Denne samsvaringen er basert på en avstand mellom vektorene i de ulike bildene. Denne avstanden kan for eksempel være Mahalanobis-, Euclidean- eller Hamming-avstand.

For at en detektor skal være god, er det visse kriterier som må oppfylles:
- Repeterbarhet
- Tidsbruk
- Skalainvarians

Repeterbarhet er en av de viktigste og detektorer blir ofte testet utifra dette. Repeterbar-

het innebærer at detektoren klarer å finne de samme egenskapspunktene gitt bildetransformasjon (rotasjon, lysendring etc). Et annet kriterium som er svært viktig er tidsbruken til metoden. Om implementeringskompleksiteten er stor vil det fort gå mye tid før detektoren finner alle egenskapspunktene. Det utvikles stadig nye applikasjoner som er avhengig av at det er en tidsbegrensning og at beregningsplassen er liten. For eksempel blir det mer vanlig at disse applikasjonene kjøres via en mobiltelefon, hvor det er begrensede beregningsressurser. Det vil også være en ulempe for 'real-time'-applikasjoner som SLAM (simultaneous localization and mapping [6]) om beregningene tar lang tid. I tillegg har størrelsen på deskriptoren noe å si for tidsbruken. Mindre størrelse tilsvarer raskere samsvar av egenskapspunkter, selv om en større størrelse vil gi mer karakteristiske egenskapspunkter. Ettersom det er blitt utviklet flere detektorer, blir det bare flere kriterier som må oppfylles for å utkonkurrere tidligere detektorer. Et kriterium som også sees på som viktig i dag er at detektoren er upåvirket av skalaendringer. Dette gir mulighet til å finne egenskapspunkter ved ulike skalaer.

1.2 Tidligere arbeid innen deteksjon av egenskapspunkter

Den mest anvendte detektoren er Harris-detektoren [7]. Harris-detektoren ble utviklet i 1988 av C.Harris og M.Stephens, og er basert på egenverdier fra Hessian-matrisen [5]. Denne detektoren er dessverre ikke upåvirket av skalaendring, som er en av de kriteriene som i dag sees på som svært viktig for en ideell detektor. Lindeberg [8] introduserte konseptet om automatisk skalaendring. Dette gav muligheten til å finne egenskapspunkter i et bilde ved ulike skalaer [5].

Etablerte ledere innen deteksjon av egenskapspunkter er i dag SURF (Speeded-up Robust Features) og SIFT (Scale Invariant Feature Transform). SURF [5] bruker determinanten til Hessian-matrisen i deteksjonen av egenskapspunkter. Den tar også i bruk integralbilder og filtrerer med boksfiltre. Deskriptoren summerer 'Haar wavelet'-responser i den bilderegionen som er funnet interessant [9] og lager en god beskrivelse av egenskapspunktet som er funnet. SIFT, utviklet i 1999 av David Lowe, er derimot utviklet som en av de bedre innen skalaendring. Den samler en mengde informasjon om intensitetsmønstrene i bildet, samtidig som den er tolerant overfor små deformasjoner og lokaliseringsfeil. Metoden bruker 'difference of gaussian'-filter for å lokalisere egenskapspunktene, løser skalaendringene ved å lage en bildepyramide hvor bildet er skalert ved ulike skalaer. I tillegg bruker den Hough-transformen til 'cluster identification' [10]. Det negative med denne metoden er at den har så høy dimensjon at den blir uforholdsmessig treg i beregningene. Metoden har blitt utvidet ved flere anledninger. Et eksempel er GLOH (Gradient Location and Orientation Histogram), som viste seg å være *enda mer karakteristisk*, men er dessverre tung beregningsmessig, noe som ikke er å foretrekke ved for eksempel online applikasjoner.

Andre detektorer som kan konkurrere med disse når det gjelder både hastighet og robusthet er BRISK (Binary Robust Invariant Scalable Keypoints) og BRIEF (Binary Robust Independent Elementary Features). BRISK [9] kan anvendes både til deteksjon, beskrivelse og 'matching' av punkter, og den bruker en FAST-basert detektor med en bit-string deskriptor fra intensitetssammenligning ved sampling av hvert område omkring egenskapspunktet. BRIEF [11] er kun laget som en deskriptor, og konkurrerer med de andre

ved at den er så rask. Denne metoden beregner direkte 'binary strings' fra bilderegioner og sammenligner intensitetene parvis langs en linje [9. fra brief]. Deskriptoren kan evalueres ved å bruke 'Hamming'-avstand og ikke L2 eller Euclidean-avstand som de fleste andre metoder benytter.

En annen metode som også har etablert seg som detektor er SUSAN (Smallest Univalue Segment Assimilating Nucleus). Denne ble først presentert av S.M. Smith og J.M. Brady i 1997, og det har blitt utviklet nye varianter av denne i senere år [12]. Selv om detektoren blir beskrevet som en hjørnedetektor, vil detektoren plukke ut andre egenskapspunkter og den plukker ut lokale bilderegioner med store gradienter i alle retninger [10]. Detektoren bruker en sirkulær maske for å finne intensitetsforskjeller i bilderegionene. Alle pikslene som har samme intensitetsverdi som senterpikselen verifiseres som USAN ('Univalue Segment Assimilating Nucleus')[13]. En terskel er satt for å skille intensitetsverdier like som senter-pikselen fra resten. Utifra hvor mange like piksler det er funnet i masken blir et hjørne eller en kant detektert.

1.3 Tidligere arbeid innen telling av sykler

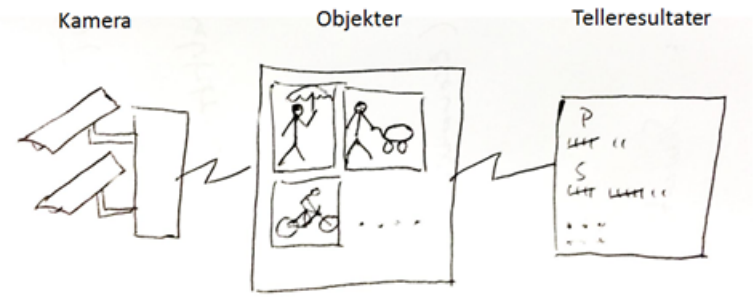
Vegvesenet står for telling av syklister i Norge. Dette er en viktig del av veiplanlegging i forhold til syklende. Tallene fra syklist-tellingen vil fortelle om sykkelstier/veier blir brukt og om de gjør slik at flere personer bestemmer seg for å bruke sykkel som transportmiddel. Dette er også viktig i forhold til miljøbevissthet. Teknologien som brukes i dag er avhengig av utformingen av veien. Metodene består av induktive sensorer som blir lagt under sykkelstien. Piezoelektriske sensorer (trykkfølsomme sensorer) er også noe som skal testes ut i nær fremtid ifølge vegvesenet.

Bildebehandling er noe vegvesenet har tenkt på når det kommer til telling av syklister. Med dette har de satt opp et 3D kameratellesystem i Drammen. Dette er et prosjekt som startet i november 2014.

Igjennom 'Nasjonal Gå strategi' har Vegvesenet lagt vekt på å få flere til å gå. Det er viktig å etablere et statistisk grunnlag over hvor mange som går i våre bysentrum. Statens

vegvesen har i samarbeid med Drammen kommune satt i gang et pilotprosjekt for å teste ut en teknologi for telling av fotgjengere og syklister.

Teknologien som testes ut er et 3D kameratellesystem levert av det svenske selskapet Viscando AB – et selskap med bakgrunn fra utvikling av våpenstyringssystemer for jagerflyindustrien.



Figur 1.1: Oversikt over 3D kameratellesystemet til Viscando AS

Løsningen bygger på prinsippene om optisk mønstergjenkjenning: det opprettes en virtuell 3D modell av den aktuelle tellesonen, og alle objektbevegelser – det vil være fotgjengere, syklister, eller andre kjøretøygrupper – vil på bakgrunn av et sett predefinerte modellindikatorer kunne registreres i systemet.

Det er viktig å poengtere at all prosessering av data foregår i sann tid – hvilket betyr at det eneste som lagres av data er passeringsantall.

Viscando holder nå på med en forbedring av klassifiseringen som bygger på kombinasjonen av geometriske/kinematiske og bildemessige egenskaper hos objektene.

Firmaet har lang erfaring med sensorer, signal- og bildeanalyse. Metoder som SIFT, SURF og HOG er blitt anvend sammen med ulike metoder for klassifisering som for eksempel SVM ('Support Vector Networks') og neurale nettverk.

1.4 Valg av detektorer til oppgave

Detektorene som er plukket ut til denne oppgaven er BRISK, SUSAN og SURF. Disse detektorene skal være rotasjons- og skalainvariante, i tillegg skal de tåle deformasjon samt finne de samme punktene gjentakende ganger. Det har blitt valgt ut tre veldig ulike detektorer for å se hvilke egenskaper en detektor må ha for å gjenkjenne objektet.

- BRISK er en binær detektor som er kjent for å være rask. I tillegg skal implementeringskompleksiteten være liten og detektoren skal være robust mot endring av lys, rotasjon og uskarphet.
- SUSAN er en enkel detektor som ikke har fått så mye oppmerksomhet, men er en detektor som er justerbar. Både en kant- og hjørnedetektor, dette gir store muligheter til spesialisering av egenskapspunkt.
- SURF er en anerkjent detektor som brukes til mange bildebehandlings-applikasjoner. Implementeringskompleksiteten skal være liten og detektoren skal være robust mot bildetransformasjoner.

1.5 Rapportens struktur

Kapittel 2 - Teori bak metodene

I dette kapittelet beskrives teorien bak de tre metodene som er valgt ut: SUSAN, SURF og BRISK. I tillegg vil Hough-transformen og bakgrunnssubtraksjon bli beskrevet i dette kapittelet.

Kapittel 3 - Verktøy

I dette kapittelet blir koden bak egenskapsdetektorene beskrevet.

Kapittel 4 - Analyse

I dette kapittelet testes SURF, SUSAN og BRISK opp mot hverandre. Metodene blir testet i forhold til hva slags egenskapspunkter de finner og hvilke punkter som vil være nyttige i denne oppgaven. Noe forarbeid i form av bildebehandling blir og anvendt her. Forslag til hvordan syklisten skal verifiseres og test av detektor på trafikkbilder blir også lagt frem i dette kapittelet.

Kapittel 5 - Resultat og diskusjon

I dette kapittelet vil resultatene fra testingen av detektorene, optimaliseringen av valgt detektor og verifiseringen av syklisten bli lagt frem. Det vil og bli diskutert rundt resultatene.

Kapittel 6 - Konklusjon

I dette kapittelet vil det bli lagt frem en konklusjon ut i fra problemstilling og resultat.

Kapittel 2

Teori

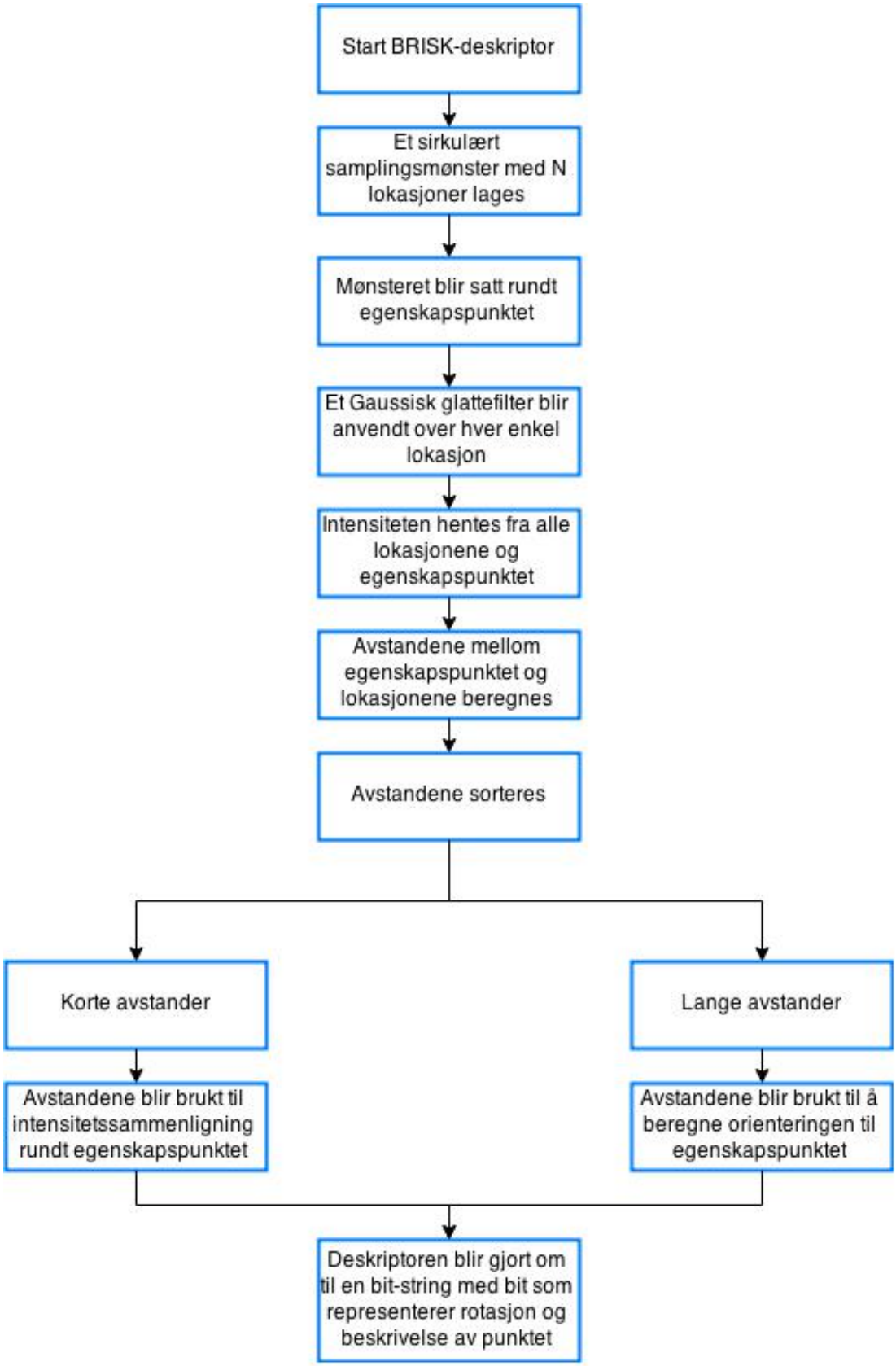
I dette kapittelet vil teorien bak BRISK-, SUSAN- og SURF-detektor bli presentert. Bildebehandlingen som er blitt gjort før analysen av detektoren blir også beskrevet i dette kapittelet.

2.1 BRISK

BRISK (Binary Robust Invariant Scalable Keypoints) ble utviklet av Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart i 2011. Hovedfokuset til denne detektoren er tidsbruk og implementeringskompleksitet. Dermed skal detektoren kunne kjøres ved applikasjoner med få beregningsressurser og online. BRISK består av en FAST-basert (Features from accelerated segment test) [14] detektor som benytter flere skalaer og en binær deskriptor laget ved hjelp av intensitetssammenligning [9]. Både detektor og deskriptor er beskrevet nærmere i delkapitlene under.

2.1.1 BRISK-detektoren

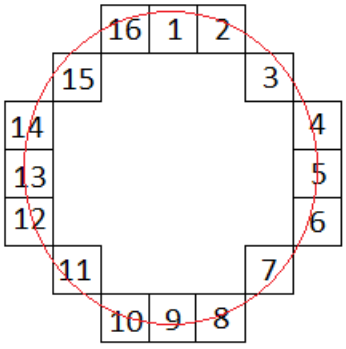
For å få et overblikk over hvordan BRISK-detektoren fungerer er det satt opp et flytdiagram (se figur 2.1).



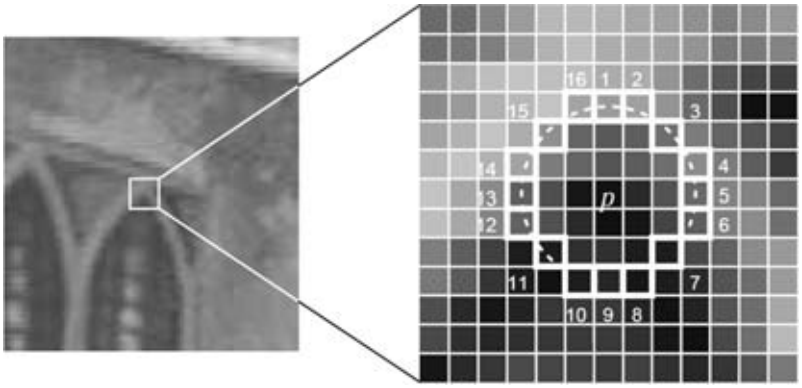
Figur 2.1: Flytdiagram for BRISK-detektoren

Et punkt blir satt som senter av et sirkulært område. Sirkelen (masken) som blir anvendt kalles Bresenham's sirkel [15]. Masken inneholder 16 piksler og for at punktet skal være definert som et egenskapspunkt må 9 av disse pikslene i tilstrekkelig grad være mørkere eller lysere enn senter-pikselen (interessepunktet). En lys piksel vil gi en høy intensitetsverdi, mens en mørk piksel vil gi en lav. I figurene 2.2 og 2.3 kan en se hvordan den sirkulære masken vil se ut.

Det er mulighet for å bruke en annen størrelse på masken, men det er i stor grad 9-16 masken som blir brukt i metodene som anvender FAST detektoren [9]. I denne sammenheng vil 16 være antall piksler i sirkelen og 9 vil være antall godkjente piksler en må ha for at et punkt skal bli karakterisert som et egenskapspunkt. Delen av detektoren som er basert på FAST kommer vi tilbake til litt senere i kapittelet.



Figur 2.2: Masken brukt i BRISK-detektoren



Figur 2.3: Sirkulær maske fra BRISK-detektoren (bilde fra [1]).

For å sjekke intensitetsdifferansen mellom interessepunktet og de 16 maskepunktene blir det laget et binært beslutningstre. Ut i fra intensitetsdifferansen til hvert enkelt maskepunkt blir de delt inn i ulike kategorier. Intensitetene kan deles inn i kategoriene nevnt i likningene i (2.1.1). De ulike kategoriene vil være:

d - mørkere pikselintensitet (lavere verdi)

s - lik pikselintensitet

b - lysere pikselintensitet (høyere verdi)

u - ikke lysere/mørkere

Deretter vil de bli sortert ved hjelp av det binære treet.

$$S_{n\to x} = \begin{cases} d, & I_{n\to x} < I_n - t & \text{(Mørkere)} \\ \bar{d}, & I_{n\to x} \not< I_n - t \wedge S'_{n\to x} = u & \text{(Ikke mørkere)} \\ s, & I_{n\to x} \not< I_n - t \wedge S'_{n\to x} = \bar{b} & \text{(Lik)} \\ s, & I_{n\to x} \not> I_n + t \wedge S'_{n\to x} = \bar{d} & \text{(Lik)} \\ \bar{b}, & I_{n\to x} \not> I_n + t \wedge S'_{n\to x} = u & \text{(Ikke lysere)} \\ b, & I_{n\to x} > I_n + t & \text{(Lysere)} \end{cases} \tag{2.1.1}$$

Her er t en terskel som blir satt etter hvor stor intensitetsdifferansen mellom interessepunktet og den aktuelle pikselen kan være før den blir kategorisert som lysere eller mørkere. Dette parameteret avgjør følsomheten for hjørnene. En høy verdi av t vil resultere i få, men svært sterke hjørner, mens en lav verdi av t gir i tillegg hjørner med jevnere gradering som resultat. I er intensiteten til en piksel og u tilsvarer at tilstanden er ukjent. Dette resulterer i et binært tre som tillater en evaluering for hver enkelt node [2].

Beslutningstreet som BRISK anvender i detektoren er inspirert av FAST-detektoren. I tillegg er det tatt noen elementer fra den utvidede versjonen av FAST som er utviklet av Mair et al., AGAST (Adaptiv and Generic corner detection based on the Accelerated Segment Test)[2].

Både FAST og AGAST, er basert på -AST ('Accelerated Segment test'). AST er laget i form av et beslutningstre. I FAST sin versjon av -AST blir beslutningstreet satt opp ved at algoritmen lærer seg distribusjonen til hjørnekonfigurasjonen ut i fra et treningssett. Treningssettet er laget som en spesiell scenestruktur.

Det er noen negative sider ved -AST slik som FAST bruker den:

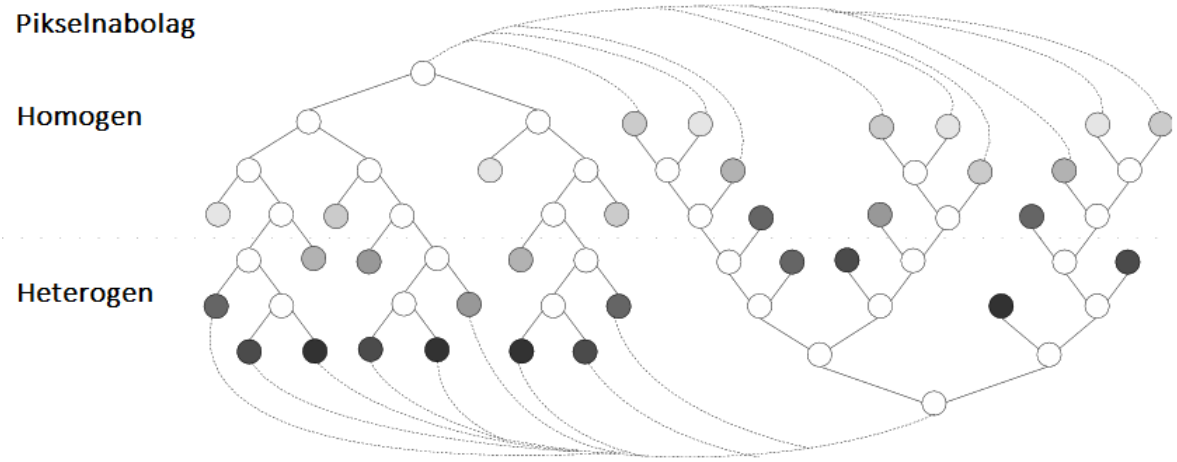
1. Hvis kamera roteres kan piksel-konfigurasjonen av et hjørne endre seg betraktelig. Dermed vil tilpasningen til en ny scenestruktur kun lønne seg i forhold til tidsbruken hvis kameraet (og/eller scenestrukturen), ikke beveger seg. Enhver bevegelse kan medføre at hjørnekonfigurasjonen blir treg å evaluere.

2. Noen hjørner vil mangle i treningssettet og dermed evalueres som 'false' når deteksjonen foregår.

3. Beslutningstreet må lære seg hjørnedistribusjonen for hver nye scenestruktur [2].

AGAST er utviklet som en forbedret versjon av AST. Metoden bygger fortsatt på det samme, men den har blitt gjort om til en adaptiv og generisk algoritme hvor den ikke må tilpasses nye omgivelser for hver nye scenestruktur og beslutningstreet er et binært tre.

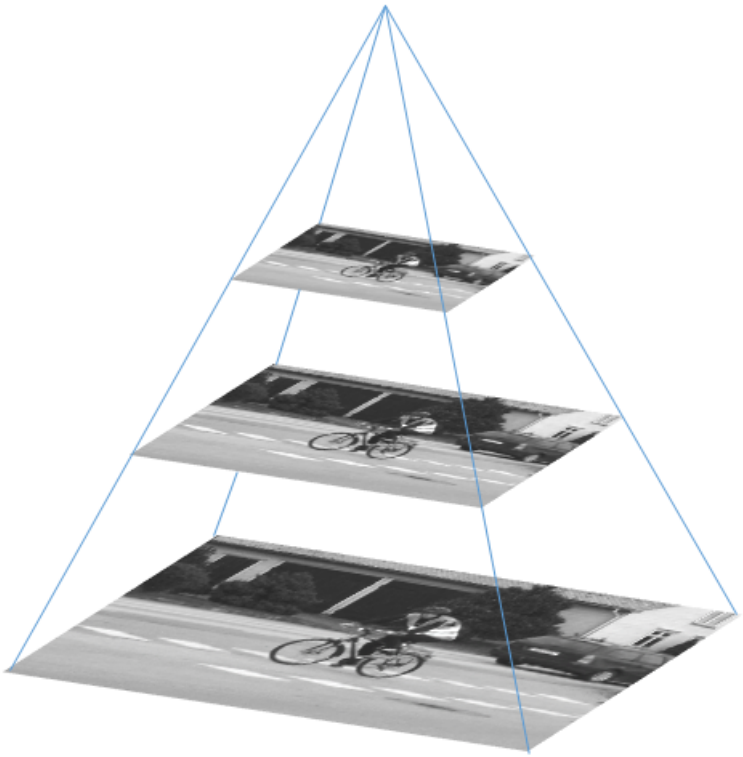
Ved å kombinere to trær kan hjørnedetektoren, AGAST, bytte tre hvis pikselnabolaget skulle endre seg. Figur 2.4 viser et slikt beslutningstre. Jo lysere farge på bladene, jo flere like piksler er det i konfigurasjonen. Det venstre treet i figuren bruker færre pikselevalueringer i et homogent pikselnabolag, mens treet til høyre er optimalisert for bilder med mye tekstur [2]. Dette treet vil kunne tilpasse seg scenestrukturen uten å måtte lage en ny konfigurasjon for hvert bilde. Det vil også være mulig å ha flere enn to beslutningstrær [2].



Figur 2.4: Et binært beslutningstre i AGAST (bildet er hentet fra[2])

2.1.1.1 BRISK som skalainvariant egenskapsdetektor

I tillegg til at detektoren er basert på FAST og AGAST, er den skalainvariant. Dette innebærer at deteksjonen av egenskapspunktene vil pågå ved flere skalaer av bildet. For å kunne lete etter egenskapspunkter ved flere skalaer bruker BRISK den gaussiske bildepyramiden (se figur 2.5).



Figur 2.5: Eksempel på en Gaussisk bildepyramide. Her vil det nederste bildet representere slik bildet ser ut ved opprinnelig skala, og de øvrige bildene vil være nedsamplet.

Den gaussiske pyramiden er satt sammen av lag hvor bildet er blitt nedsamplet mellom lagene. I denne sammenheng kalles de ulike lagene for oktaver og midt-oktaver. Bildepyramiden vil inneholde n oktaver c_i og n midt-oktaver d_i. Fra en oktav til en annen vil bildet bli nedsamplet med 2, i tillegg vil det være lag mellom dette som er nedsamplet med 1.5 (midt-oktav). $t(c_i) = 2^i$ for oktavene og $t(d_i) = 2^i \cdot 1.5$ for midt-oktavene [9]. Her representerer t en skala og ikke en terskel som tidligere. Før bildet nedsamples vil et gaussisk glattefilter bli brukt på bildet.

Når bildet er blitt samplet n ganger (n tilsier så mange nedsamplinger man vil ha. det er vanlig at det er 4 i denne sammenheng) vil bildepyramiden være ferdig.

Et eksempel på hvordan et bilde ser ut når det blir nedsamplet med 2 vises i figur 2.6.

Figur 2.6: Bildet viser en sekvens med 4 bilder, hvor bildet har blitt nedsamplet med 2 mellom hvert bilde. Første bildet (til venstre) viser det originale bildet. Bildene som har blitt nedsamplet er i dette tilfellet skalert opp, derfor ser bildene like store ut.

Detektoren vil lete etter egenskapspunkter for alle skalaene av bildet. Deretter vil disse punktene sammenlignes med hverandre. For å kunne sammenligne egenskapspunktene funnet ved de ulike skalaene bruker FAST en verdi s ('salience') som sier noe om hvor fremtredende et punkt er. Her er hensikten å finne ut ved hvilken skala det mest fremtredende egenskapspunktet befinner seg i. Ved hvert egenskapspunkt sjekkes samme punkt i ett oktavnivå over og under. Det brukes et område på 3x3 piksler. Ut i fra hvilken av disse tre oktavene som inneholder punktet med høyest verdi av s blir det avgjort hvilken skala som passer akkurat dette punktet.

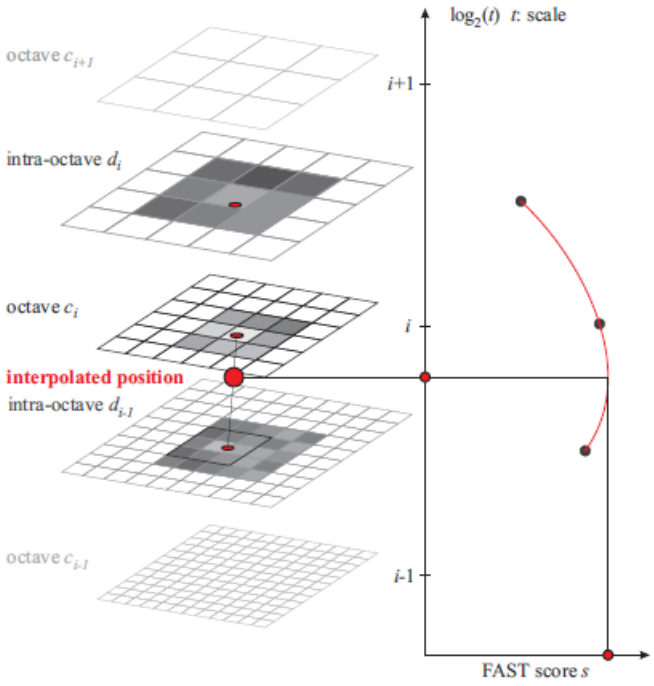
Det er kun maksverdien, s, en er interessert i fra pikselområdene. For å få maksimalverdien alene brukes 'non-maxima suppression'. Dette innebærer at området utelukker alle verdiene innenfor området som er lavere enn maksverdien. Dermed sitter en igjen med kun ett egenskapspunkt i det gitte området. Verdiene som er lavere enn maksimalverdien s vil bli satt til null og blir ikke tatt høyde for videre i deteksjonen.

Deretter vil maksimalverdiene tilpasses en '2D quadratic function' (minste kvadraters metode). Minste kvadraters metode estimerer punktene ved å finne en sammenheng mellom data en har og en enkel annengradsfunksjon som tilpasser disse dataene best mulig. Metoden er todimensjonal, noe som betyr at man bruker 2 variabler, skala og maksimalverdi.

Maksimalverdiene og skalaene til oktavnivået over og under det aktuelle punktet vil også bli tilpasset minste kvadraters metode. Deretter blir disse annengradsfunksjonene inter-

polert ved å bruke en endimensjonal parabel. Skalaen for det aktuelle interessepunktet blir bestemt ut i fra den høyeste verdien av s gitt funksjonen som er laget [9].

For å vise hvordan dette fungerer er figuren 2.7 lagt til. Her kan en se hvordan fremtredenheten av punktet forandrer seg mellom de forskjellige skalaene og parabelen som er tilpasset annengradsfunksjonene.



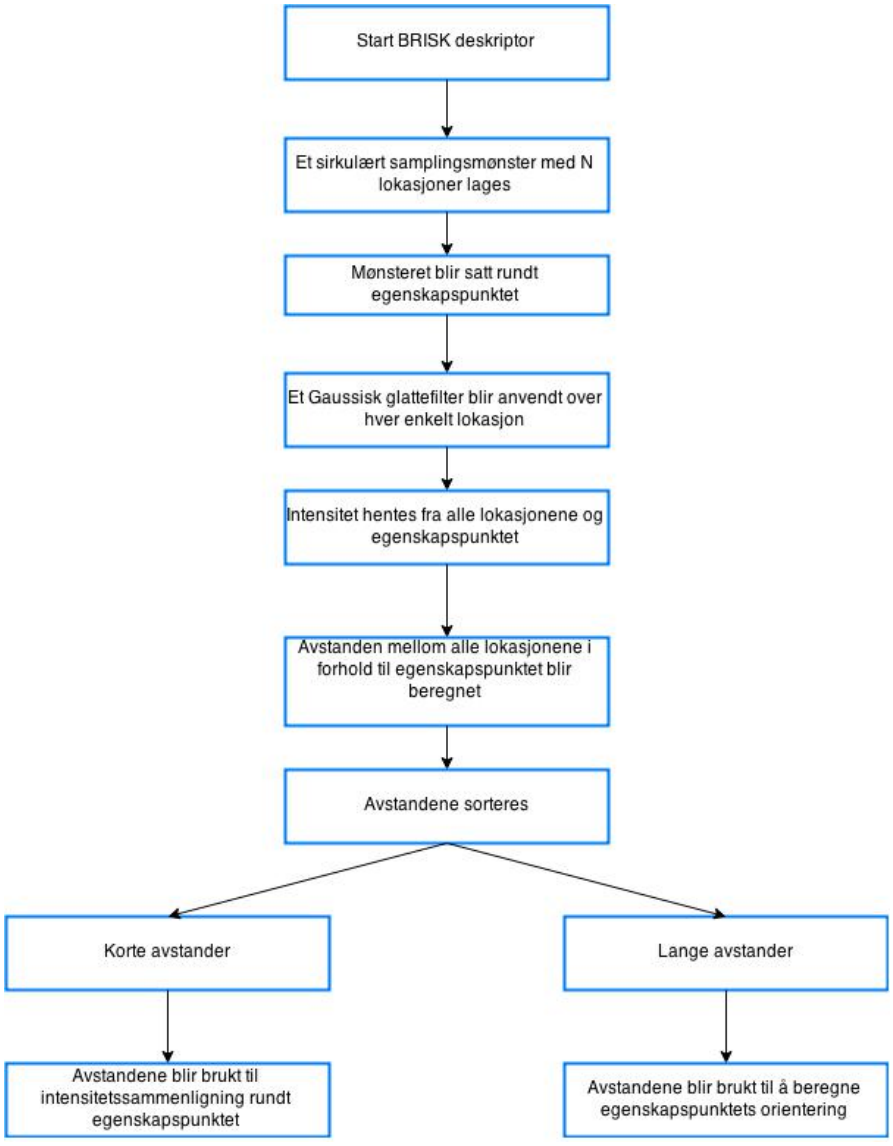
Figur 2.7: Bildet er hentet fra [9] og viser hvordan de forskjellige oktavene av bildet er satt opp. Til høyre vises det hvordan verdien for fremtredenhet, s, forandrer seg ettersom oktavene blir evaluert

Tilslutt re-interpoleres lokasjonen til egenskapspunktet ut i fra maksimalverdien nærmest til skalaen som er valgt ut for dette punktet.

Søket etter maksverdiene i c_0 (det originale bildet) vil være litt annerledes enn i de andre oktavene. Her vil det være en FAST-maske som er noe mindre, 5-8 (masken består av 8 piksler og 5 av disse må være kvalifisert som tilstrekkelig mørkere eller lysere enn interessepunktet). Dette er for at det skal være mulig å opprettholde en virtuell midt-oktav d_{-1} under c_0. Verdien$s$ i midt-oktaven d_{-1} må ikke være større enn verdien den har i c_0 [9].

2.1.2 BRISK-deskriptoren

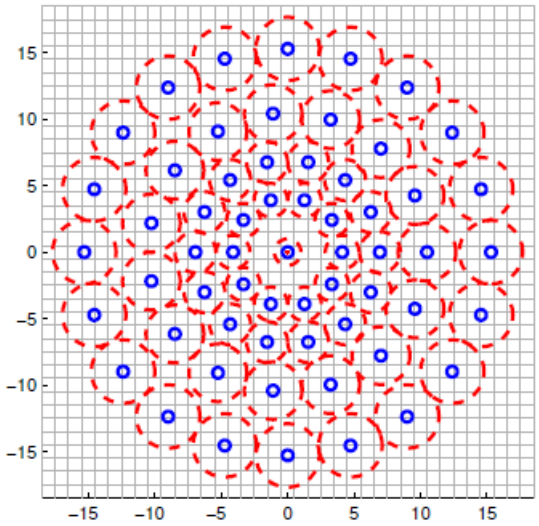
BRISK-deskriptoren er en binær deskriptor. For å bygge en binær deskriptor sammenlignes intensiteten til punkter parvis rundt egenskapspunktene, det trengs kun to pikselpunkter per egenskapspunkt. Denne fremstillingen av en deskriptor er beregningsmessig enkel, men allikevel representerer den egenskapspunktene godt [16].
En oversikt over hvordan BRISK deskriptoren fungerer vises i figur 2.8.



Figur 2.8: Flytdiagram for BRISK-deskriptoren

Et sett av egenskapspunkter er funnet ved detektoren og lagret med lokasjon og passende skala. Denne informasjonen brukes videre for å beskrive alle egenskapspunktene som er funnet.

Et samplingsmønster blir laget og brukt på hvert enkelt egenskapspunkt, p_i. Samplingsmønsteret er satt sammen av N lokasjoner rundt egenskapspunktet. Disse lokasjonene er satt i konsentriske sirkler om punktet og avstanden mellom lokasjonene er lik. Figur 2.9 viser hvordan punktene er lokalisert i forhold til egenskapspunktet.



Figur 2.9: Bildet er klippet ut fra [9] og viser hvordan samplingsmønsteret til BRISK-deskriptoren ser ut. Her med en skala som tilsvarer t=1 og N=60 lokasjoner om egenskapspunktet.

For å unngå aliasing under samplingen av intensitetene brukes et Gaussisk glattefilter ('smoothing') med et standardavvik, σ_i, over hver enkelt lokasjon. Standardavviket vil være proporsjonalt med avstanden mellom punktene på den aktuelle sirkelen. De røde sirklene rundt punktene viser det normale standardavviket på det Gaussiske filteret i figur 2.9.

Deretter vil samplingpunkt-par bli funnet. Dette er punkter man kan sammenligne med egenskapspunktet og lage en god beskrivelse med. For å finne antall samplingpunkt-par, S_p brukes denne formelen $S_p = \frac{N(N-1)}{2}$. Hvis en her tar samplingsmønsteret til figuren

2.9, hvor det er 60 lokasjoner rundt egenskapspunktet, vil antall samplingspunktpar bli:

$$S_p = \frac{60(60-1)}{2} = 1770 \tag{2.1.2}$$

Deretter skal de lokale gradientene i hvert egenskapspunkt beregnes. For å kunne beregne disse må vi ha intensitetsverdiene til punktene. Intensitetsverdiene er glattet ut av det Gaussiske glattefilteret og tilsvarer $I(p_i, \sigma_i)$ og $I(p_j, \sigma_j)$. Deretter estimeres de lokale gradientene:

$$g(p_i, p_j) = (p_i - p_j)\frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{||p_j - p_i||^2} \tag{2.1.3}$$

Alle samplingpunkt-parene vil deles i to kategorier etter hvor lang avstand det er mellom hvert enkelt par. De to kategoriene er:
Kort avstand, S, og lang avstand L. Disse er satt til $\delta_{maks} = 9.75t$ og $\delta_{min} = 13.67t$. t representerer skalaen punktet er funnet ved.

Kriteriene for de to kategoriene er som følger:

$$S = ||p_j - p_i|| < \delta_{maks}, \qquad L = ||p_j - p_i|| > \delta_{min} \tag{2.1.4}$$

De lange avstandene blir brukt til å lage orienteringen til egenskapspunktet, mens de korte brukes til intensitetssammenligning omkring punktet. Det er de korte avstandene som bygger deskriptoren[16].

Vektoren g vil representere retningen (orienteringen) til det gitte egenskapspunktet. Denne blir beregnet ut i fra parene innenfor kategorien L. Formelen for denne vektoren lyder som følger:

$$g = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \frac{1}{L} \sum_{(p_i, p_j)\epsilon L} g(p_i, p_j)$$

Ved dannelsen av BRISK deskriptoren gjelder samplingsmønsteret rotert ved $\alpha = arctan2(g_y, g_x)$ rundt egenskapspunktet, k. Dette blir hentet fra vektoren g. Bit-vektor deskriptoren, d_k,

inneholder intensitetssammenligningen til de korte avstandene til punktparene (p_i^α, p_j^α) ϵ S. Dermed vil hver enkelt bit i bit-vektoren (gitt rotasjonen som er beregnet) respondere til:

$$b = \begin{cases} 1, & I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, & ellers \end{cases} \tag{2.1.5}$$

Det blir til slutt laget en binær bit-string av bit'ene, b, beregnet over. Stringen vil inneholde både egenskapspunktets rotasjon og intensitetsbeskrivelse. Denne vil være enkel og rask å matche opp mot andre versjoner av bildet (lysendring, rotasjon osv.).

2.2 Metoden SUSAN

SUSAN (Small Univalue Segment Assimilating Nucleus) detektoren ble utviklet av S. M. Smith og J. M. Brady i 1997. Detektoren baserer seg på å assosiere hver enkel piksel med et lokalt område av bildet (bilderegion) som har lik intensitet som pikselen. Deretter vil egenskapsdetektoren minimalisere denne bilderegionen. SUSAN blir sett på som en effektiv detektor som er robust mot lokal støy og rask i evaluering av piksler [17]. En stor fordel ved denne detektoren er at det ikke blir brukt noen bilde-deriverte, noe som gjør den mer robust mot støy. En mer detaljert beskrivelse av detektoren er å finne i underkapitlene.

2.2.1 SUSAN-detektoren

SUSAN-detektoren bruker sirkulære masker med en gitt radius over alle punkter i et bilde. Et regelverk (i form av likninger) avgjør om et hjørne eller en kant er funnet. Den mest anvendte radiusen på masken er 3.4 piksler (dette gir en maske som inneholder 37 piksler) [17].

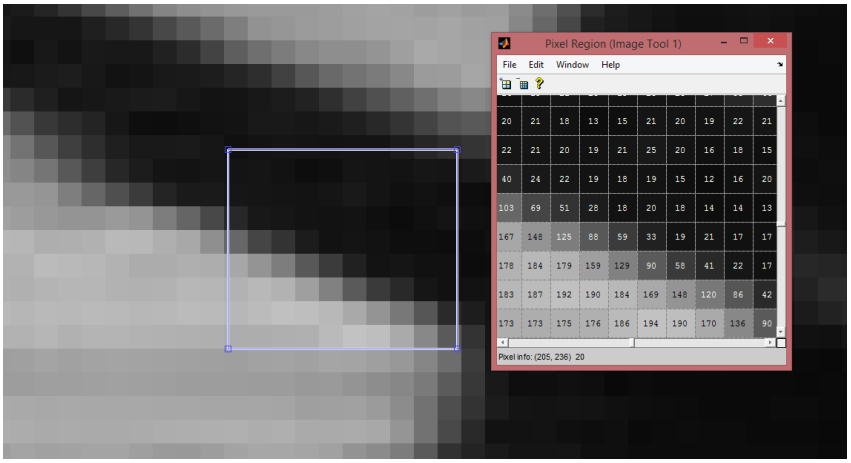
For hvert punkt blir intensitetene til pikslene innen masken sammenlignet med maskens senter-piksel (som vil bli referert til som kjernen videre i rapporten). Hvis pikselen har en intensitet lik kjernen vil den bli kategorisert som en del av et bildeobjekt. For å kunne avgjøre ulikheten i intensitet mellom kjerne og vilkårlig piksel i masken har en terskel, t. Dette betyr at om differansen er større enn terskelen, t, vil ikke pikselen være en del av objektet.

Alle pikslene innen masken som tilfredsstiller denne betingelsen blir kalt USAN ('Univalue Segment Assimilating Nucleus')[18]. En enkel likning for denne sammenligningen er vist i likning (2.2.1). USAN vil beskrive bildets struktur godt. Det er disse områdene som benyttes når kantene og to-dimensjonale egenskaper (for eksempel hjørner) skal bli funnet.

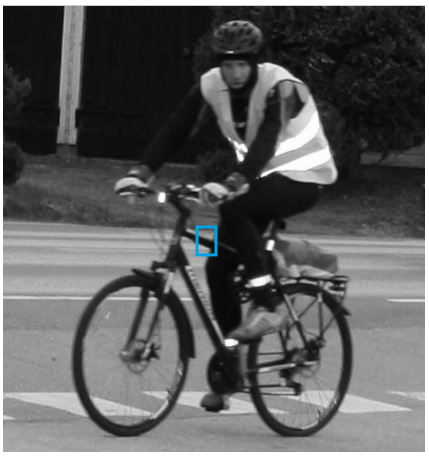
$$c(\vec{r}, \vec{r_0}) = \begin{cases} 1, & \text{hvis } |I(\vec{r}) - I(\vec{r_0})| \leq t. \\ 0, & \text{hvis } |I(\vec{r}) - I(\vec{r_0})| > t, \end{cases} \tag{2.2.1}$$

I likningen vil $I(\vec{r_0})$ representere intensiteten til kjernen, r_0. Intensiteten til det punktet kjernen sammenlignes med vil være $I(\vec{r})$. $c(\vec{r}, \vec{r_0})$ er funksjonen som bestemmer om et punkt ansees som en del av USAN eller ei, hvor r vil være punktet som sammenlignes med kjernen. Terskel t er satt for å avgjøre om ulikheten mellom piksel, r, og kjernen,r_0, er for stor for å være en del av USAN. Den vil også avgjøre hvor mye støy som vil bli ignorert. Om denne terskelen settes stor, vil hjørne- og kantdeteksjonen være mer nøyaktig og utelukke uskarpe hjørner/kanter som ellers ville blitt karakterisert som et hjørne/en kant.

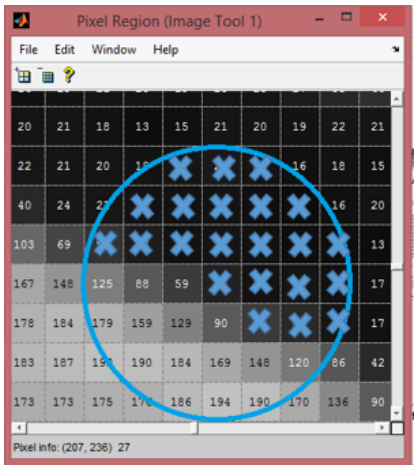
Et eksempel på hvordan det vil se ut når intensitetsverdiene innen masken sammenlignes og en skal verifisere USAN er vist i figur 2.10. Bilde b) viser det opprinnelige bildet, hvor en rektangulær boks viser området av bildet som skal analyseres. a) viser hvordan utklippet ser ut i forstørret tilstand og til høyre er masken visualisert over gråverdiene innen utklippet. c) viser hvilke piksler som har blitt verifisert som USAN. Terskelen t er her satt til 25 gråverdier, og kjernen hadde en intensitetsverdi på 33. Dermed blir øverste grenseverdi 58 og nederste blir 8. Alle intensitetsverdier som havner utenfor disse grensene vil ikke være en del av USAN i denne masken.



(a) Et område i bildet er plukket ut og til høyre vises gråverdiene til pikslene det gjelder.



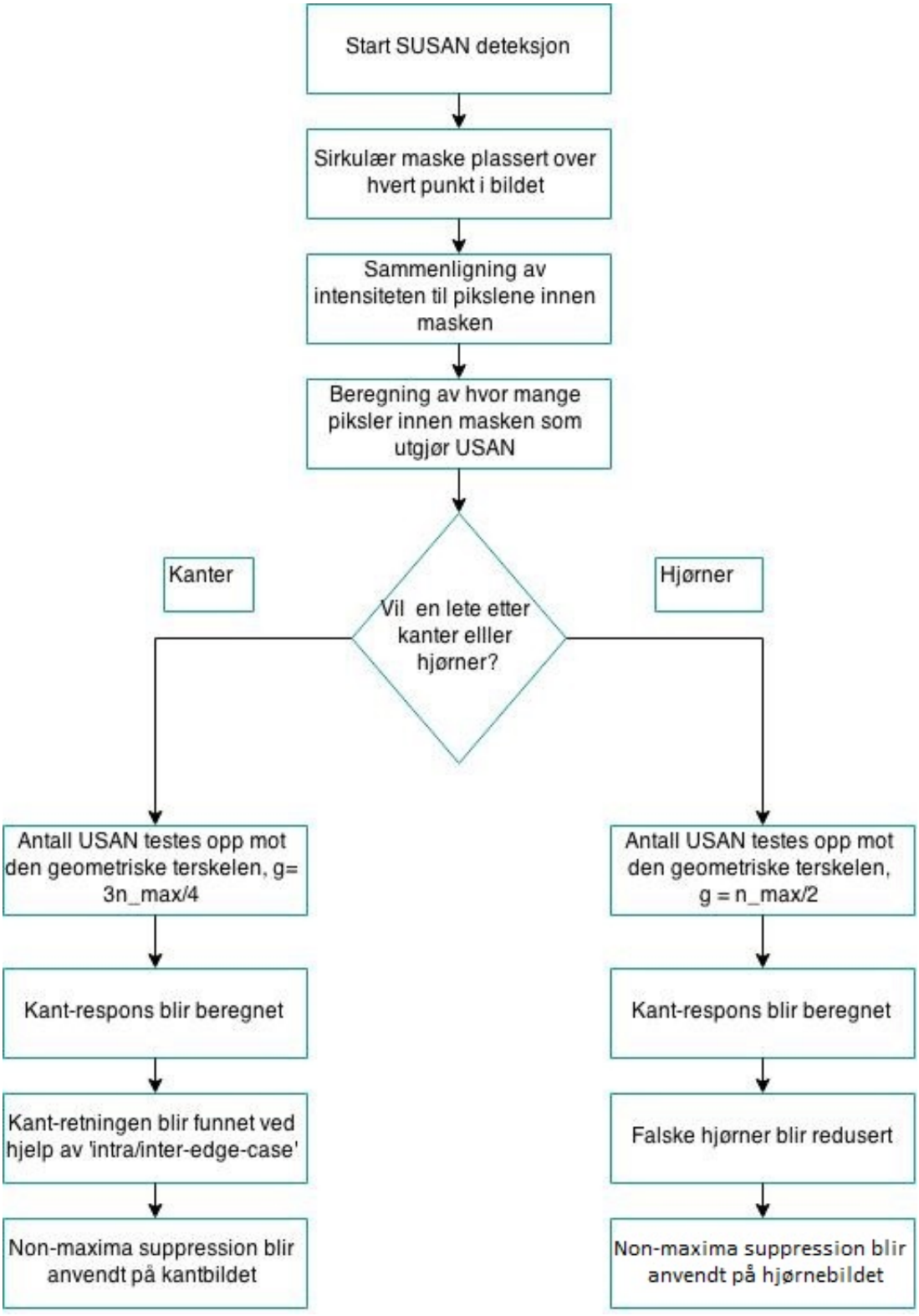
(b) Det opprinnelige bildet



(c) USAN-godkjente piksler representert med blå kryss.

Figur 2.10: Sammenligning av intensiteter og deretter verifisering av USAN-areal.

For å få en bedre oversikt over SUSAN-detektoren før en går inn i detaljene er et flytdiagram lagt til i figur 2.11.



Figur 2.11: Flytdiagram for SUSAN-detektoren

Likningen (2.2.1) gir gode resultater, men en mer stabil metode for å beregne $c(\vec{r}, \vec{r_0})$ vil være likning (2.2.2).

$$c(\vec{r}, \vec{r_0}) = e^{(\frac{I(\vec{r}) - I(\vec{r_0})}{t})^6} \tag{2.2.2}$$

Denne likningen gir en piksels intensitet mulighet til å kunne variere noe uten at dette påvirker $c(\vec{r}, \vec{r_0})$. Anvendelsen av de to ulike USAN-likningene er vist i figur 2.13. At likning (2.2.2) er opphøyd i sjette kan vises å være optimalt teoretisk. Denne formen gir en balanse mellom god stabilitet ved terskel t og den opprinnelige funksjonen (2.2.1). Dette er ekvivalent med å oppfylle kriteriet som sier at det skal være et minimum antall falske negative og falske positive. Dette kriteriet er formulert i følgende uttrykk:

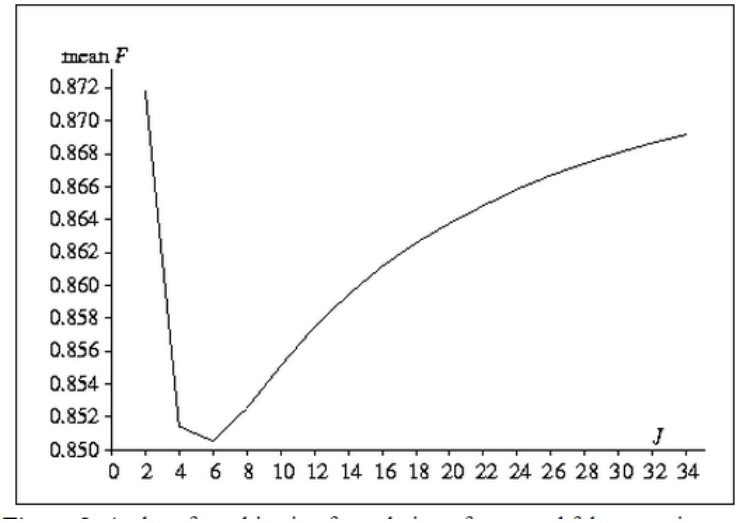
$$F(d, t, s) = \frac{\sqrt{var(R_S)} + \sqrt{var(R_N)}}{< R_S > - < R_N >} \tag{2.2.3}$$

Hvor F er proporsjonal med antall falske positive og falske negative som blir rapportert. s er støyens standardavvik, R_N er SUSAN's kantrespons uten kanter og R_S er SUSAN's kantrespons når masken er plassert over en kant med styrke d.

Verdien til F vil variere i forhold til verdiene: d, t og s. Hvis en her setter t til 1 og varierer de to andre variablene vil det dekke alle tilfellene. Intervallene til d og s er valgt til å være 0-10 og 0-$\sqrt{2s}$ henholdsvis. SUSAN-detektoren er basert på:
$\Delta = I(\vec{r}) - I(\vec{r_0})$
At s har fått denne maksimalverdien kommer av at hvis bildets støy er Gaussisk med standardavvik lik s vil støyen til Δ også være Gaussisk med standardavvik lik $\delta_\Delta = \sqrt{2s}$.

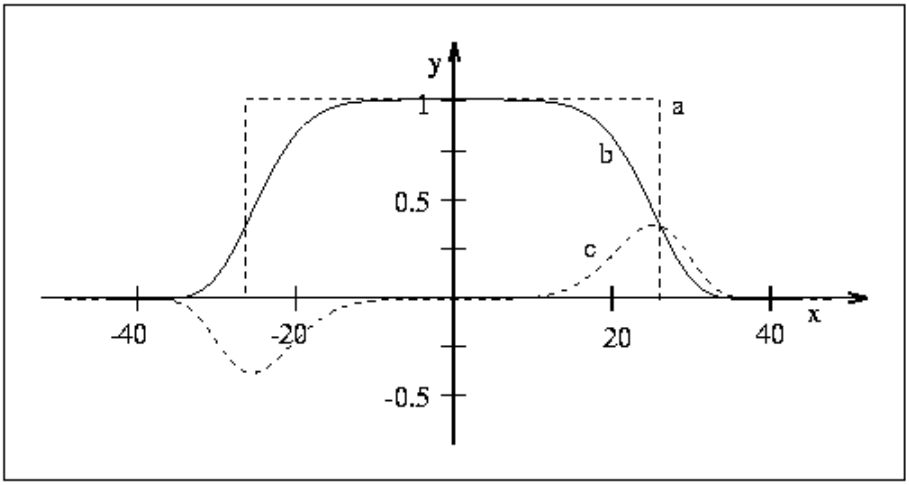
For å evaluere F, d, t og s er det nødvendig å finne forventningsverdiene og variansene av R_S og R_N. Disse likningene og resultatet finnes i artikkelen [17].

Intervallene over d og δ_Δ blir beregnet og det endelige resultatet av F blir funnet. Resultatet av F i forhold til J er plottet i figur 2.12 og viser tydelig at å opphøye funksjonen i sjette vil optimalisere SUSAN-filteret [17].



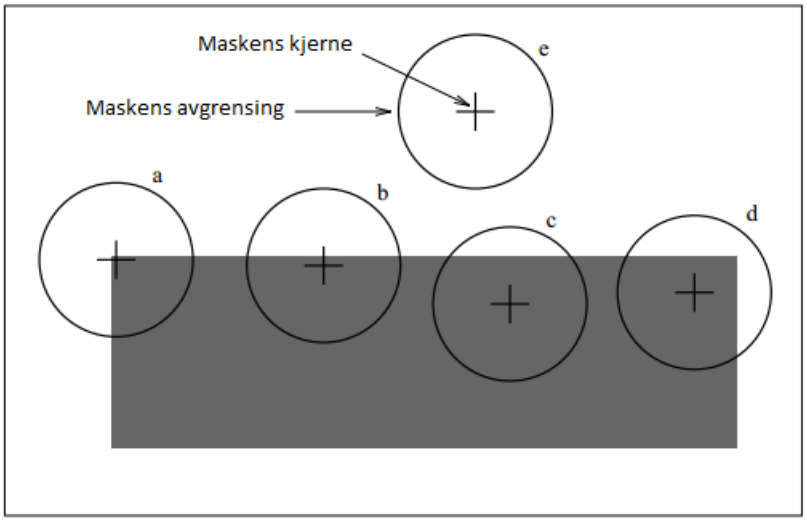
Figur 2.12: Viser verdien F mot verdier av J. Viser tydelig at å opphøye i sjette vil optimalisere filteret [17].

For å sammenligne den enkle funksjonen (2.2.1) og den gaussiske funksjonen (2.2.2) er de plottet i samme graf, se figur 2.13.



Figur 2.13: a) Den opprinnelige sammenligningsfunksjonen (y aksen, ingen enhet) mot intensitetsforskjellen til pikslene (x aksen, i gråtonivåer). I dette tilfellet er terskelen t satt til 27 (27 gråtoner blir dermed godkjente som USAN). b) Her er den gaussiske funksjonen (2.2.2) brukt, og en ser at avgrensingen av gråtoner blir noe mer avrundet. Figuren er hentet fra [17]

Her ser en at a representerer funksjonen (2.2.1) og b representerer den gaussiske funksjonen (2.2.2). Figuren viser hvor stor terskelen for gråtoner som blir tatt med i USAN er. Funksjon a verifiserer 27 nivåer av gråtoner som en del av USAN, det betyr at terskelen t er satt til 27 i dette tilfellet. Den gaussiske funksjonen, b, ser en at gir en mer avrundet avgrensning av gråtoner i forhold til a. Funksjon b vil gi et mer stabilt resultat. c vil representere den deriverte i punktet hvor terskelavgrensningen blir satt ved funksjon a.



Figur 2.14: SUSAN-masken plassert over 5 ulike punkter i et bilde. Bildet er hentet fra [17].

Figur 2.14 viser hvordan SUSAN-masken beveger seg rundt i et bilde. Her er bildeobjektet et enkelt rektangel med en hvit bakgrunn. De fem sirklene representerer masken ved fem ulike punkter. Krysset inni hver maske representerer kjernen, og sirkelen viser avgrensningen av masken.

Maskene vil ha ulik mengde USAN-areal ut i fra hvor masken befinner seg i bildet. Arealet varierer med tanke på hvor mange verifiserte USAN-piksler det er i masken. Gitt figur 2.14 ser man at maske e vil ha et maksimalt areal med USAN. Her er hele masken fylt med piksler med samme intensitet. b og c vil ha et mindre USAN-areal, grunnet at maskene er i kanten av det rektangulære objektet og fylt med to ulike gråverdier. Maskene med minst USAN-areal vil være det som blir kategorisert som hjørner. Dette tilsvarer maske a.

Likningen (2.2.4) viser hvor mange USAN-piksler hver enkelt maske inneholder.

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \tag{2.2.4}$$

n vil tilsvare antall piksler som tilfredsstiller USAN betingelsen innenfor en maske. Deretter vil n bli sammenlignet med g, den geometriske terskelen. g avhenger av om man er ute etter hjørner eller kanter. Herfra vil algoritmen for SUSAN-detektoren være litt ulik avhengig av egenskapene en er mest interessert i å finne.

2.2.1.1 SUSAN kantdetektor

Ved kantdeteksjon blir den geometriske terskelen, g, satt til $\frac{3n_{maks}}{4}$, hvor n_{maks} tilsvarer antall piksler i masken (ofte 37). Terskelen er beregnet ut i fra forventningsverdien av kantresponsen med kun støy. Kantresponsen er gitt ved likning (2.2.5).

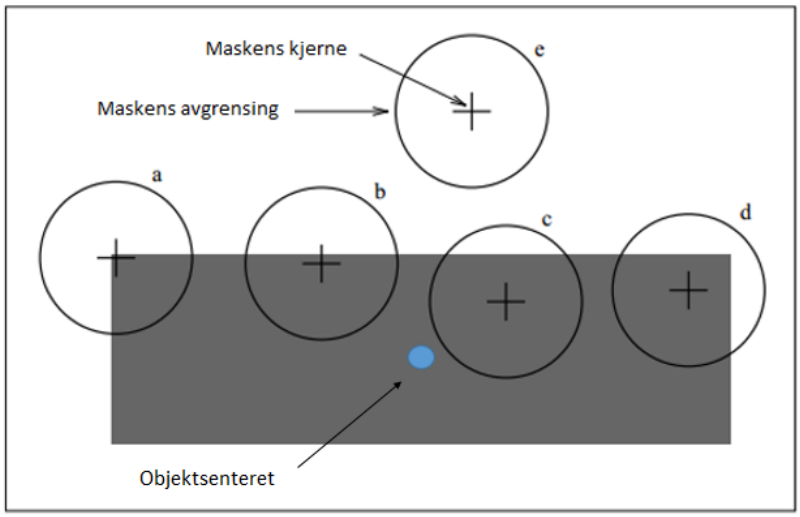
$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}), & \text{if } n(\vec{r_0}) < g. \\ 0, & \text{ellers,} \end{cases} \tag{2.2.5}$$

Dette er en enkel formulering av SUSAN-prinsippet, jo mindre USAN-arealet er, jo større kantrespons [17].

Deretter vil en beregne retningene til de kantene som er funnet. En er avhengig av 'non-maxima-suppression' senere i deteksjonen og i den beregningen inneholder kantenes retninger.
Kant-retningen assosiert med et punkt i bildet som har en kantverdi (et punkt som er blitt karakterisert som en kant) blir funnet ved å analysere USAN. USAN gir to tilfeller, 'intra-piksel' eller 'inter-piksel' [17].

Hvis USAN-arealet (i piksler) er mindre enn maske-diameteren (i piksler) vil man bruke 'intra-piksel'. Hvis USAN-arealet derimot er større enn denne grensen vil en finne tyngdepunktet for USAN, og bruke denne til å beregne retningen til kanten ved hjelp av 'Inter piksel'.

Med tyngdepunktet menes punktet som vil være midtpunktet av objektet som USAN befinner seg i (se figur 2.15).

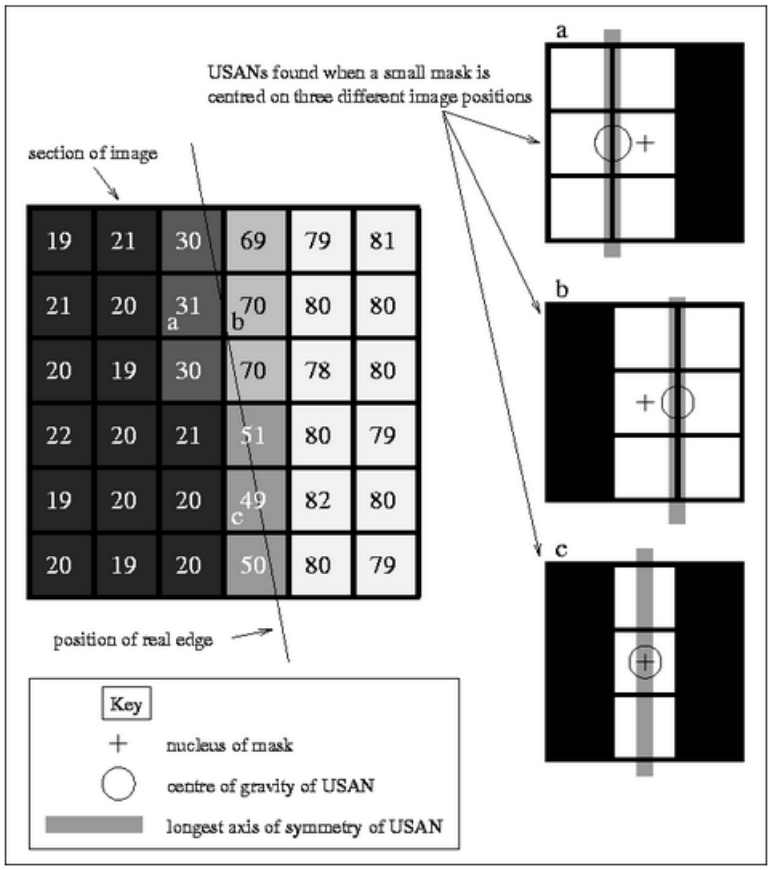


Figur 2.15: Den blå sirkelen representerer tyngdepunktet i det rektangulære USAN-området merket med grått.

Ved 'Inter-piksel' vil vektoren mellom tyngdepunktet, $\vec{r}$, av USAN og kjernen av masken være normalt på den lokale kantretningen. Et eksempel på dette er punktet gitt i figur 2.16 *a* og *b*. Tyngdepunktet kan beregnes ved likningen (2.2.6).

$$\bar{\vec{r}}(\vec{r_0}) = \frac{\sum_{\vec{r}} \vec{r} c(\vec{r}, \vec{r_0})}{\sum_{\vec{r}} c(\vec{r}, \vec{r_0})} \tag{2.2.6}$$

Hvis en beregner tyngdepunktet til å ligge mindre enn én piksel unna kjernen til masken, er det mer naturlig å bruke 'Intra-piksel'. Grunnen til dette er at kjernen ligger så nær tyngdepunktet at området vil ligge inne i bildeobjektet. Et eksempel på dette er *c* i figur 2.16.



Figur 2.16: Figuren viser tre ulike posisjoneringer av masken over et USAN-område. Bildet er hentet fra [17].

Ved 'Intra-piksel' vil USAN være formet som en tynn linje i kantretningen. Kantretningen kan dermed regnes ut ved å finne den lengste symmetriaksen. Dette blir estimert ved å bruke likningene (2.2.7), (2.2.8) og (2.2.9).

$$\overline{(x-x_0)^2}(\vec{r_0}) = \sum_{\vec{r}} (x-x_0)^2 c(\vec{r}, \vec{r_0}) \tag{2.2.7}$$

$$\overline{(y-y_0)^2}(\vec{r_0}) = \sum_{\vec{r}} (y-y_0)^2 c(\vec{r}, \vec{r_0}) \tag{2.2.8}$$

$$\overline{(x-x_0)(y-y_0)}(\vec{r_0}) = \sum_{\vec{r}} (x-x_0)(y-y_0) c(\vec{r}, \vec{r_0}) \tag{2.2.9}$$

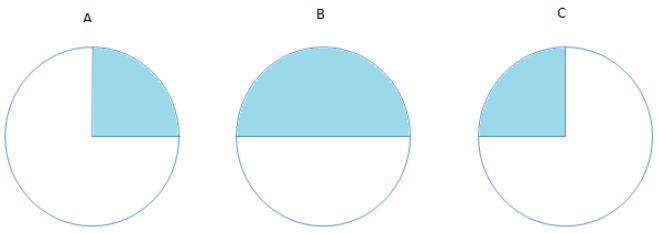
Forholdet mellom $\overline{(x-x_0)^2}$ og $\overline{(y-y_0)^2}$ blir brukt for å fastslå retningen til kanten.

$\overline{(x - x_0)(y - y_0)}$ blir brukt for å finne ut om en diagonal kant har positive eller negative helninger.

Deretter vil det bli brukt 'non-maxima suppression' på kantbildet. Dette er for å forhindre minimumsverdier (kun vinkelrett på kanten) i å blir kategorisert som kanter.
Kantbildet kan bli 'binary thinned'. Dette er en prosess hvor kantene blir tynnet ut for å forsikre seg om at kantene oppfyller kravene for antall-piksel-naboer som må være koblet sammen. Det er også for å forsikre seg om at man har fjernet alle pikslene som ikke hører til kanten og for eventuelt å erstatte kantpunkter som ble tatt bort ved 'non-maxima suppression' ved en feil [17]. Et regelsett er implementert for 'binary thinning' som vil gi et godt binært bilde. Disse reglene er beskrevet i [19].

2.2.1.2 SUSAN-hjørnedetektor

Det er ikke mye som skiller SUSAN-kantdetektor fra hjørnedetektoren. I de første stegene av algoritmen fungerer de to detektorene helt likt. Alle pikslene innen en maske blir sammenlignet med maskens kjerne, og likningene (2.2.4) og (2.2.2) presentert ved kantdeteksjonen er også brukt her. n blir også sammenlignet med den geometriske terskelen g. Men i dette tilfellet vil terskelen være annerledes enn for kantdetektoren. Hvis kjernen ligger i et hjørne slik som A i figur 2.17 vil USAN-arealet være under halvparten av arealet til masken, og vil være ett lokalt minimum. En kan sette g til å være halve størrelsen av n_{maks} fordi en rett kant alltid vil dekke mer enn halvparten av masken, i og med at den inkluderer kjernen.



Figur 2.17: Delene av maskene som er markert med blått viser områdene av maskene som utgjør USAN-arealet.

Dermed blir likningen for terskel g som i likning (2.2.10).

$$g = \frac{n_{maks}}{2} \tag{2.2.10}$$

Likning (2.2.5) blir fortsatt brukt for å finne kantresponsen i bildet, men med den nye verdien av terskel g [17].

$$R(\vec{r_0}) = \begin{cases} \frac{n_{maks}}{2} - n(\vec{r_0}), & \text{if } n(\vec{r_0}) < g. \\ 0, & \text{ellers,} \end{cases} \tag{2.2.11}$$

Ved bruk av SUSAN-detektoren kan falske hjørner bli detektert. Dette skjer for eksempel ved at uskarpe kanter mellom to bildeobjekter blir kategorisert feil. Derfor er det utviklet to ulike metoder for å kvitte seg med denne type feil.

1. Den første metoden går ut på å finne tyngdepunktet i USAN. Deretter blir avstanden fra kjernen til tyngdepunktet beregnet. Hvis masken med USAN er kategorisert som et hjørne vil tyngdepunktet ha lang avstand fra kjernen, mens en tynn linje som går igjennom kjernen vil ha en kort avstand fra tyngdepunktet til kjernen. Dermed vil en kunne utelukke kanter som vil bli funnet midt inni et bildeobjekt.

2. Den andre metoden tvinger kontinuitet i USAN områdene. Alle pikslene i masken som ligger i en rett linje som peker utover i forhold til kjernen og i retning av tyngdepunktet til USAN må være en del av USAN for at hjørnet skal bli detektert. Dette tvinger USAN til å ha en form for ensartethet ('uniformity') og reduserer de falske hjørneresponsene [17].

Deretter søker en over hele bildet, med en kvadratisk maske på 5x5 piksler, etter lokale maksimalverdier. Maksimalverdiene vil være verdiene som representerer hjørnene.

2.2.2 SUSAN som en skalainvariant egenskapsdetektor

SUSAN-detektoren som en skalainvariant detektor vises ved grafer som viser kantlokasjonene mot maskestørrelsen (plott av enkle horisontale linjer fra kantbildet mot maskestørrelse gir vertikale linjer. Dette er utvilsomt en ønskelig egenskap, siden det betyr at nøyaktighet ikke avhenger av maskestørrelse. Dette forventes: minimum av USAN-areal når det nærmer seg en kant inntreffer uavhengig av maskestørrelsen [17].

2.2.3 SUSAN-detektoren brukt i denne oppgaven

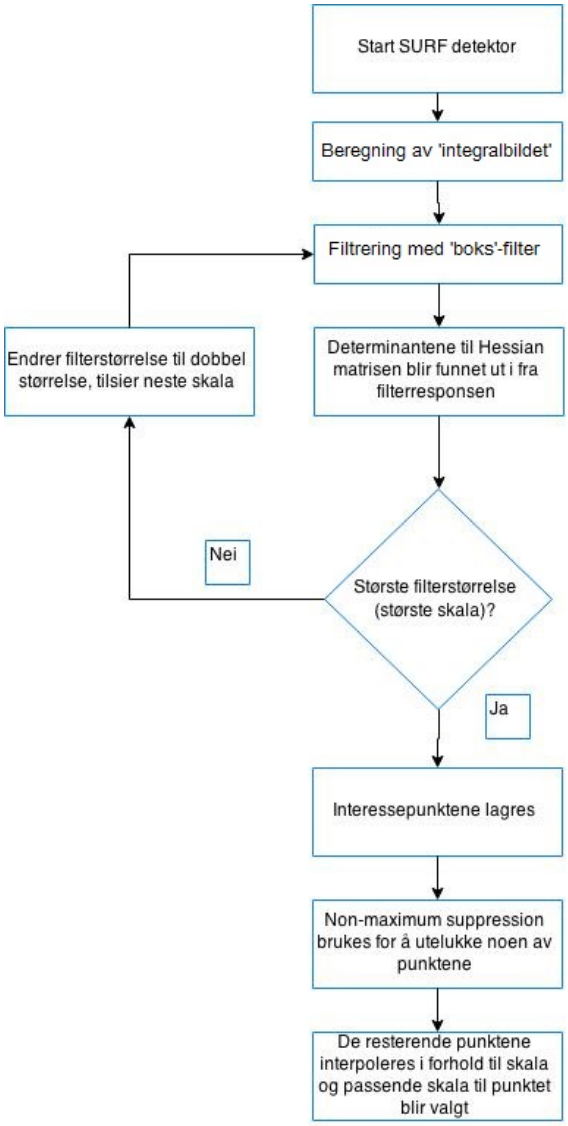
I denne rapporten er en interessert i å finne egenskapspunkter ved hjulet av en sykkel. Dermed vil jeg bruke en kombinasjon av kant- og hjørnedetektoren. Den geometriske terskelen er satt til: $g = 0.625 \cdot n_{maks}$. Dette er midt mellom de to egenskapene. For å finne hjulet er en avhengig av å finne egenskapspunktene ved kantene til hjulene. Hjørnedetektoren vil her bli for snever, mens kantdetektoren vil finne veldig mange punkter (flere vil bli regnet som støy). Dermed er det konkludert med at det må brukes en SUSAN-detektor som kan finne punkter som ligger noe mellom dette.

2.3 Metoden SURF

SURF (Speeded-Up Robust Features) ble utviklet og presentert av Herbert Bay, Andreas Ess, Tinne Tuytelaars og Luc Van Gool i 2007 [5]. SURF-detektoren baserer seg på integralbilder (beskrevet senere i kapittelet), filtrering og beregninger ved hjelp av Hessian-matrisen. Deskriptoren er basert på Haar Wavelet responser. SURF detekterer blob-egenskaper i bilder og blir anvendt ved en rekke applikasjoner innen bildebehandling. Detektor og deskriptor er beskrevet mer detaljert i delkapitlene under.

2.3.1 SURF-detektoren

Før en går inn i detaljene på SURF-detektoren er det lagt til et flytdiagram i figur 2.18. som beskriver dens virkemåte.



Figur 2.18: Flytdiagram for SURF-detektoren.

SURF-detektoren baserer seg på en enkel versjon av Hessian-matrisen [20]. Hessian-matrisen er en kvadratisk matrise som inneholder 2. ordens partiell deriverte av funksjonen. Matrisen beskriver lokale krumninger til en funksjon med flere variable. Hvert komponent i Hessian-matrisen blir beskrevet som likning (2.3.1) [20].

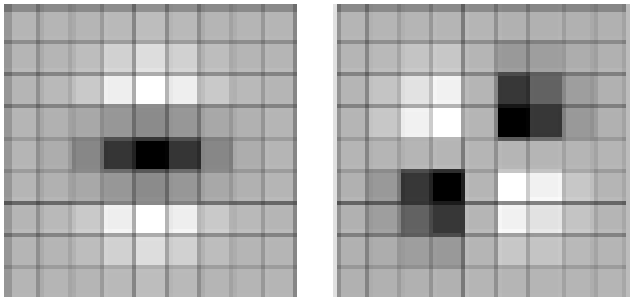
$$H_{i,j} = \frac{\delta^2 f}{\delta x_i \delta x_j} \tag{2.3.1}$$

Detektoren bruker Hessian-matrisen grunnet nøyaktigheten detektoren får. Den finner blob-strukturer i bildet hvor determinanten til matrisen er størst i verdi. Determinanten

brukes også til valg av passende skala for egenskapspunktene. Gitt et punkt X = (x,y) i et bilde I, er Hessian-matrisen $H(X, \sigma)$ i punkt X med skala σ definert som følgende:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yy}(X, \sigma) \end{bmatrix}$$

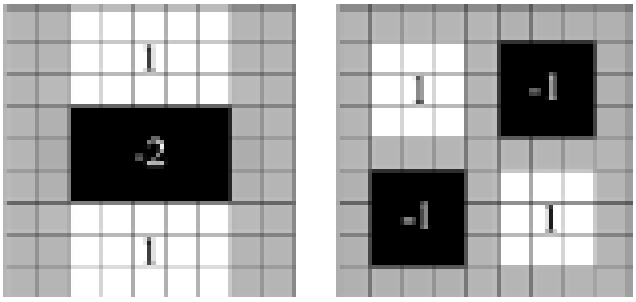
Hvor $L_{xx}(x, \sigma)$ representerer konvolusjonen mellom bildet, I og den Gaussiske 2. ordens masken, $\frac{d^2}{dx^2}g(\sigma)$,og g er en gaussfordeling med standardavvik σ. Tilsvarende for $L_{xy}(x, \sigma)$ og $L_{yy}(x, \sigma)$.



Figur 2.19: Venstre: Gaussisk 2. ordens deriverte maske i y-retning, L_{yy}. Høyre: Gaussisk 2. ordens deriverte maske i xy-retning, L_{xy}. Figur er hentet fra [5]

Figur 2.19 viser den Gaussiske 2. ordens deriverte masken i y-retningen (venstre), som tilsvarer L_{yy}, og i xy-retningen (høyre), som tilsvarer L_{xy}. Områdene som er grå tilsvarer verdi 0, svarte -1 og hvite 1.

I SURF-detektoren er det gjort en tilnærmelse til den Gaussiske 2. ordens deriverte masken (disse vil være diskretisert og klippet til). Disse er blitt gjort om til boksfiltre. Boksfiltrene som SURF bruker er vist i figur 2.20.



Figur 2.20: Boksfilter

Figuren over representerer et 9x9 (piksler) boksfilter som tilsvarer en Gaussisk maske med $\sigma = 1.2$. Dette vil representere den laveste skalaen.

De nye maskene blir kalt D_{xx}, D_{xy}, og D_{yy}. Det blir også lagt til en vekting til de rektangulære regionene for å effektivisere beregningene. Dermed blir determinanten til Hessian-matrisen:

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.3.2}$$

Vektingen w av filterresponsen blir brukt til å balansere uttrykket for Hessian determinanten. Dette brukes for å bevare energien mellom de Gaussiske maskene og de tilnærmede Gaussiske maskene. Likningen for w er vist i (2.3.3).

$$w = \frac{|L_{xy}(1.2)|_F|D_{yy}(9)|_F}{|L_{yy}(1.2)|_F|D_{xy}(9)|_F} = 0.912.. \simeq 0.9 \tag{2.3.3}$$

$|X|_F$ tilsvarer Frobenius norm. Teoretisk sett vil vektingen endre seg avhengig av skalaen. Men i praksis sees w på som konstant, 0.9.
Frobenius norm er en matrise norm. For mxn-matrisen X er Frobenius norm definert som kvadratroten til summen av absoluttverdiene til elementene: $|X| = \sqrt{\Sigma_{i=1}^{m}\Sigma_{j=1}^{n}|x_{ij}|^2}$.

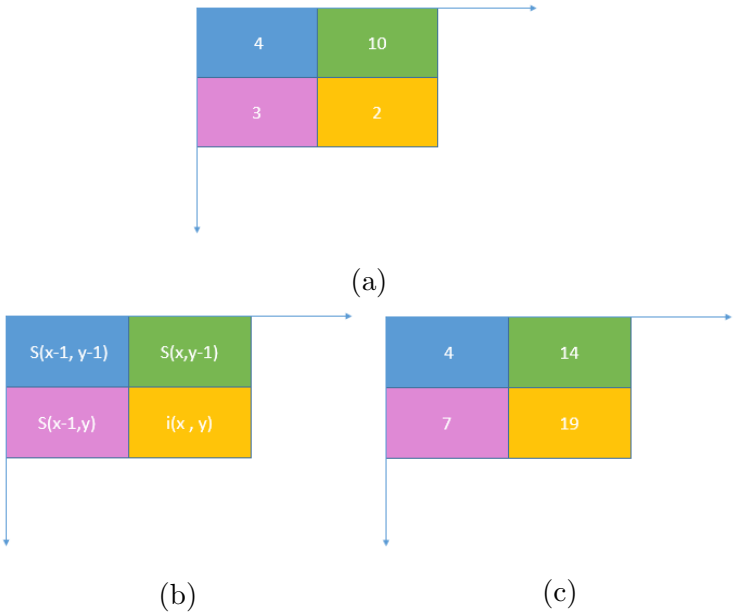
Ved å bruke boksfiltrene er det mulig å beregne konvolusjonen effektivt med en vilkårlig størrelse på masken ved å utnytte integralbilder. Et annet uttrykk for integralbilder er

'Summed Area Table'. Dette ble presentert av Viola og Jones i 2001 [21]. Etter dette er det flere egenskapsdetektorer som bruker integralbilder i implementeringen.

Et integralbilde representerer summen av gråtoneverdiene til pikslene innenfor bilderammen. Hvis du har et punkt $X = (x, y)^T$ vil verdien i dette punktet representere alle verdiene fra punkt (0,0) i bildet til X. Integralbildet kan videre brukes til å beregne gjennomsnittsintensiteten i et gitt bilde.

$$I_\Sigma(x) = \sum_{i=0}^{i<x} \sum_{j=0}^{j<y} I(i, j) \tag{2.3.4}$$

Et eksempel på hvordan disse verdiene summeres i rektangelet er vist i figuren 2.21.



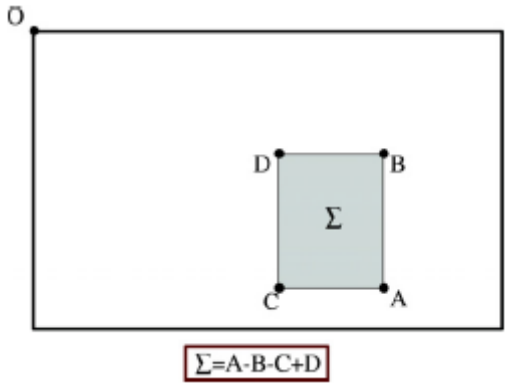
Figur 2.21: Figuren viser hvordan verdiene blir summert sammen i et integralbilde.

a) viser 4 verdier fra bildet som man vil lage et integralbilde av. Ved å bruke likningen 2.3.5 kan de nye verdiene i integralbildet beregnes.

$$s[x, y] = i[x, y] + s[x - 1, y] + s[x, y - 1] - s[x - 1, y - 1] \tag{2.3.5}$$

De resulterende verdiene av det summerte området ser en i figur 2.21 c).

Med utgangspunkt i integralbildet trengs kun tre addisjoner for å få gjennomsnittsintensiteten av det rektangulære området. Dette er vist i figur 2.22.

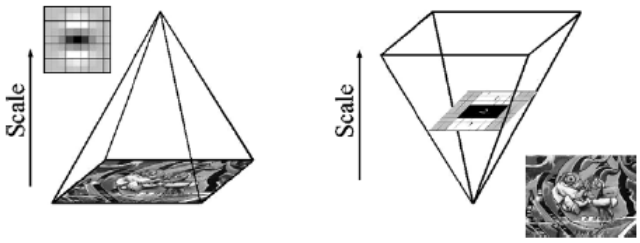


Figur 2.22: Figuren viser hvordan en finner gjennomsnittsintensiteten ved kun 3 addisjoner på et integralbilde. Figuren er hentet fra [5]

Grunnet at kalkulasjonen av filterresponsen (intensitetsgjennomsnittet) kun består av 3 addisjoner vil ikke kalkulasjonstiden være avhengig av størrelsen på rektangelet. Dette er en fordel for SURF som bruker store filtre [5].

2.3.2 SURF som en skalainvariant egenskapsdetektor

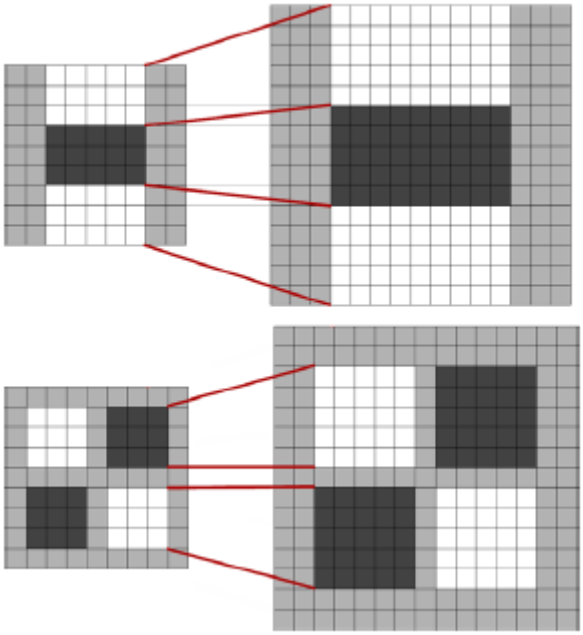
Ved andre skalainvariante egenskapsdetektorer bruker en som regel bildepyramiden i figur 2.5 hvor en nedsampler bildet mellom deteksjonen av egenskapspunkter for å oppnå skalainvarians. Grunnet SURF sin bruk av boksfiltre og integralbilder vil det ikke være behov for å iterativt anvende samme filteret over utgangsbildet fra forrige bildefiltrering. I stedet kan man anvende boksfiltre av ulike størrelser direkte på det opprinnelige bildet. Dermed er skalaendringen løst ved å oppskalere (up-scaling) filterstørrelsen i stedet for å iterativt redusere bildestørrelsen [5]. Et bilde som illustrerer dette er vist i figur 2.23.



Figur 2.23: Høyre: Her vises bildepyramiden, hvor skalaen endres ved å nedsample bildet med 2 for hver oktav. Venstre: Her vises SURF sin metode ved å endre filterstørrelse for hver nye skala.

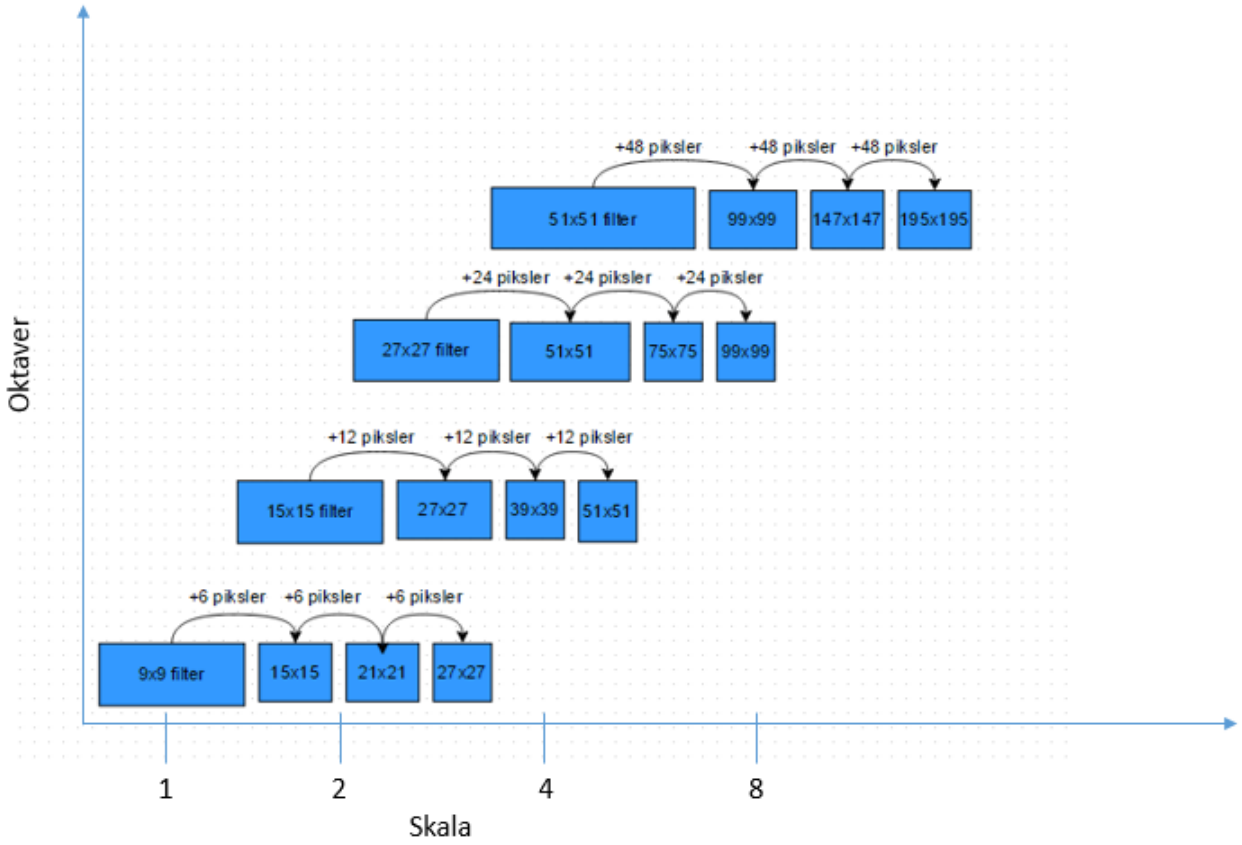
Utgangsverdien for 9x9 boksfilteret, blir brukt som den første skalaen, referert til som skala $s = 1.2$. Skalaen tilsvarer bruk av Gaussisk filter med $\sigma = 1.2$.De neste lagene oppnås ved å gradvis forstørre masken. Fordelen med dette er beregningsmessig effektivitet. Det kommer av at bildet ikke trenger å bli nedsamplet mellom lagene, og det vil derfor ikke forekomme aliasing.

Som nevnt tidligere består bildepyramiden av flere oktaver (lag). Mellom oktavene blir bildet glattet ut med et Gaussisk filter og nedskalert med 2. I SURF sin metode vil en oktav i bildepyramiden representere en serie av filterresponser innhentet ved å konvolvere det opprinnelige bildet med et filter med økende størrelse.
Et eksempel på hvordan økningen i filterstørrelse vil se ut er vist i figur 2.24.



Figur 2.24: Filter D_{yy}(øverst) og D_{xy}(nederst) for to suksessfulle skalanivåer (9x9 og 15x15). Bildet er hentet fra [5].

For to suksessfulle nivåer innen en oktav, må vi øke filterstørrelsen med minimum 2 piksler (1 piksel i hver retning) for å holde senter-pikselen (interessepunktet) på samme sted. Dette resulterer i en total økning av 6 piksler på maskestørrelsen. Dermed vil første oktav representeres av filtrene 9x9, 15x15, 21x21 og 27x27. Her ser en at økningen mellom filtrene er 6 piksler. Figur 2.25 viser filterstørrelsene ved de ulike skalaene. Filtrene vil overlappe hverandre.



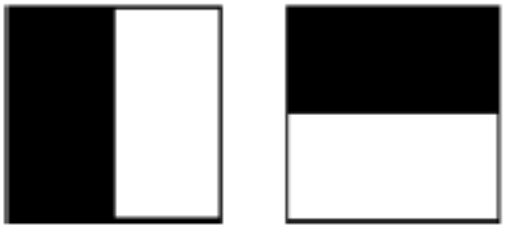
Figur 2.25: Her vises filtrene brukt i de ulike oktavene.

Frobenius norm vil være den samme for alle filterstørrelsene, disse er allerede skala-normaliserte. Dermed trengs det ingen videre filter-vekting.

Alle egenskapspunktene som er funnet ved Hessian determinanten i de ulike skalaene blir lagret. For å lokalisere egenskapspunktene over ulike skalaer blir det brukt 'non-maxima suppression' på egenskapspunktene som er funnet. Deretter blir maksimalverdien av Hessian-matrisens determinant interpolert i forhold til skala og 'image space' med en metode introdusert av Brown og Lowe [22]. Interpolasjonen mellom de ulike skalaene er spesielt viktig i dette tilfellet grunnet den store forskjellen i størrelse mellom de første oktavene.

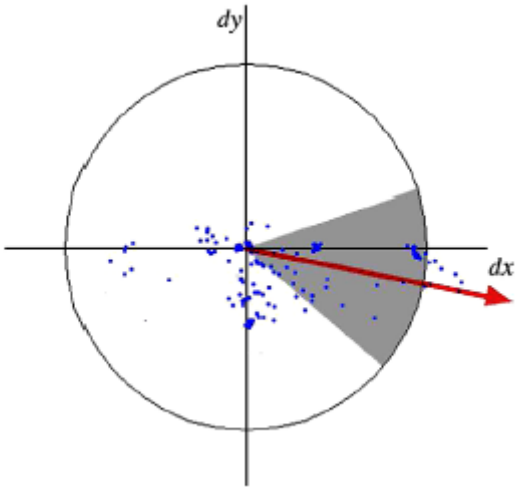
2.3.3 SURF deskriptoren

I tillegg til at SURF-detektoren er skalainvariant er den også invariant mot rotasjonsendringer. For å finne rotasjonen til hvert enkelt egenskapspunkt blir det brukt Haar Wavelet filterresponser. Haar wawelet filteret er vist i figur 2.26.



Figur 2.26: Denne figuren viser Haar Wavelet filtrene som beregner responsene i x- og y-retning. De svarte delene tilsvarer en verdi på -1 og de hvite +1.

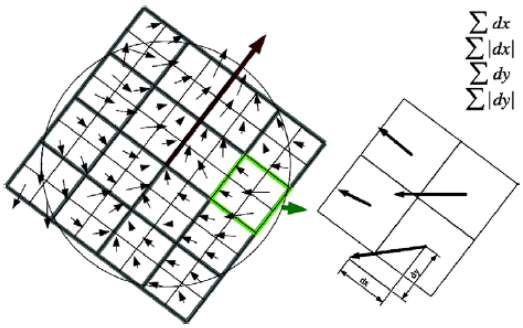
Haar Wavelet responsene i x- og y-retningene innen et sirkulært område med en radius på $6{\cdot}s$ (hvor s er skalaen egenskapspunktet ble funnet ved) blir funnet. Størrelsene på wavelet'ene er avhengige av skalaen og er satt til å være $4{\cdot}s$. Kun 6 beregninger skal til for å beregne wavelet-responsen i x- og y-retning ved en vilkårlig skala. Når Haar wavelet responsene er beregnet og veid med Gaussian ($\sigma = 2s$), som er plassert på interessepunktet, blir responsene representert som punkter langs x-aksen om det er en horisontal respons og langs y-aksen om det er en vertikal respons. Slik som vist i figur 2.27.



Figur 2.27: Sektor-vindu som roterer med vinkelen og har utstrekning $\frac{\pi}{3}$.

Deretter blir den dominante orienteringen estimert ved å beregne sum av alle responsene innen et sektor-vindu som roterer med vinkelen og har utstrekning $\frac{\pi}{3}$. Se figur 2.27.De horisontale og vertikale responsene blir summert. De to summene blir den lokale retningsvektoren for punktet [5].

Det skal også lages en beskrivelse egenskapspunktet. Denne beskrivelsen vil gå ut i fra retningen som allerede er beregnet. Deskriptoren starter med et kvadratisk område rundt egenskapspunktet, størrelsen på dette vinduet vil være $20s$. Dette interesseområdet blir delt inn i 4x4 sub-områder som blir beskrevet av hver sin wavelet-respons i x- og y-retning, referert til som dx og dy. Se figur 2.28 for visualisering av vektorene. Det samme filteret som ble brukt til orienteringen er anvendt ved deskriptoren 2.26, men her er vektingen av det Gaussiske filteret $\sigma = 3.3s$ (s representerer skalaen egenskapspunktet er funnet ved). Dette er for å unngå deformasjoner og lokaliseringsfeil.



Figur 2.28: Vektorene beregnet for de ulike subområdene og deres lengde innenfor det kvadratiske området. Her er subområdene vist som 2x2 i stedet for 4x4. Her ser en også rotasjonen som er beregnet i forkant. Figuren er hentet fra[5].

For å få informasjon om polariteten av intensitetsendringene, henter vi også ut absoluttverdiene av responsene, $|d_y|$ og $|d_x|$. Dermed har hvert subområde hver sin deskriptor vektor $v = (\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|)$. En vektor blir beregnet ut i fra hvert sub-område. Tilsammen vil disse 16 vektorene (en fra hvert subområde) sammensatt representere området rundt interessepunktet.

Tilslutt blir deskriptoren normalisert for å opprettholde invarians mot variasjoner i kontrasten.

2.4 Sirkeldeteksjon

Sirkeldeteksjonen skal brukes til å finne sykkelhjulene ut de gitte egenskapspunktene. Det er viktig at metoden som blir brukt har mulighet til å lete over et intervall av sirkelradier i og med at hjulet på sykkelen kan variere. Syklisten vil ha mulighet til å sykle i ulik avstand fra kameraet og det er viktig at hjulet fortsatt blir funnet selv om størrelsen ikke er eksakt lik. Her har en valgt å bruke Hough-transfomens sirkeldeteksjon. Transformen tar for seg et intervall med radier og er kjent for sin robusthet.

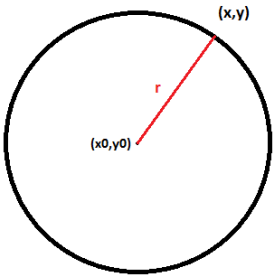
2.4.1 Hough transformen

Hough-transformen ble oppfunnet av Paul Hough som tok patent på ideen i 1962, men det er den generaliserte Hough-transformen, utviklet av Richard Duda og Peter Hart i 1972 som hovedsakelig blir brukt i dag. Hough-transformen går ut på å tilpasse en struktur til et sett av egenskapspunkter. For å gruppere egenskapspunktene til å passe samme struktur registreres alle strukturene i hvert enkelt egenskapspunkt og deretter ser en etter strukturer som 'får mange stemmer'. Sirkelen med flest stemmer beskriver best strukturen. En tar hvert egenskapspunkt og avgjør hvor mange strukturer som går gjennom dette punktet [23].

Hough-transformen er en allsidig metode og kan brukes til å finne alle mulige former i et bilde. I dette tilfellet vil strukturen være en sirkel.

2.4.1.1 Sirkeldeteksjon med Hough-transformen

En sirkel er beskrevet med tre parametre: x_0, y_0 og r. Dermed får vi en tredimensjonal Hough-matrise. Hvis en skal lete etter sirkler med ubegrenset størrelse blir dette en veldig krevende operasjon som krever mye minne. Dermed avgrenses søket ved å lage et intervall som inneholder de interessante radiene. En ulempe vil da være om man spesifiserer feil radius og ikke vil finne de i bildet.



Figur 2.29: Sirkel.

Formelen for en sirkel er: $(x - x_0)^2 + (y - y_0)^2 = r^2$. Før en kan finne sirkler i et bilde må egenskapspunktene blir funnet. For hvert egenskapspunkt(x,y) blir parameterne x_0, y_0 og

r beregnet for alle mulige sirkler som tilfredsstiller formelen for en sirkel gjennom punktet. Dermed blir Hough-matrisen tredimensjonal. Hvert egenskapspunkt i bildet blir en sirkel for hver mulige verdi av radius i parameter-planet, og hvert punkt i parameter-planet tilsvarer en sirkel i bildet. Dersom vi har to punkter, vil disse tilsvare to sirkler for hver radius i Hough-planet. Krysningspunktet til sirklene tilsvarer parameterne til den sirkelen som går innom begge punktene. Hough-matrisen blir inkrementert for hvert egenskapspunkt i bildet som blir berørt av sirkelen som svarer til ett av punktene i parameter-planet. Inkrementeringen skjer selvfølgelig i cellen som tilsvarer parameterne til den aktuelle sirkelen. Dette kalles en stemmeprosess. Når Hough-transformen har gått gjennom alle egenskapspunktene og samlet opp alle stemmene fra hvert egenskapspunkt lagres de i Hough-matrisen. Deretter letes det etter lokale maksimumsverdier i Hough-matrisen. Alle lokale maksimumsverdier i matrisen representerer parameterne til de sirklene som går gjennom flest egenskapspunkter. Pseudokode for transformen:

```
for alle x
    for alle y
      hvis (x,y) er et punkt i kanten
        for alle x0
          for alle y0
            r = sqrt((x - x0)^+(y - y0)^2)
            inkrementer cellen i Hough-matrisen tilsvarende punktet (x0; y0; r)
          end
        end
      end
    end
end
```

Den direkte implementeringen av pseudokoden er lite effektiv og minne-krevende. Koden som blir brukt for Hough-transformen er hentet fra MATLAB's File Exchange sider [24], laget av David Young.

2.5 Bakgrunnssubtraksjon

I mange applikasjoner vises objekter på en stabil bakgrunn. Et eksempel som ofte brukes er deteksjon av deler på et transportbånd. Et annet eksempel er et kamera lokalisert i et rom. Alt i rommet som ikke ligner på rommet (bakgrunnen) vil være interessant. I slike typer applikasjoner kan det være nyttig å bruke segmentering ved å subtrahere et estimat av bakgrunnen fra bildet for å lete etter store absoluttverdier i resultatet [23]. Dette gir opphav til begrepet bakgrunnssubtraksjon.

I denne oppgaven er bakgrunnssubtraksjon tatt med som en forbehandling av bildet før egenskapsdetektorene anvendes. Ved å subtrahere bakgrunnen segmenteres det ønskede objektet, og det er færre punkter for detektoren å lete over. Ved bakgrunnssubtraksjon er det viktig at en har en rekke bilder av bakgrunnen. Grunnen til dette er at bakgrunnen endres sakte over tid og det vil derfor være vesentlig å ta utgangspunkt i et gjennomsnitt av flere bilder før en subtraherer bakgrunnen.

Det finnes flere metoder for bakgrunnssubtraksjon. Dette er grunnet ulike typer bakgrunner som skal fjernes og eventuell støy i bildene mellom de ulike bilderammene. I denne oppgaven anvendes metoden 'moving average'.

Ved en bakgrunn som endrer seg mye (et eksempel på forstyrrelser er vær og vind) kan det være en fordel å bruke en metode som vektlegger piksler som endres mer enn andre. Ideelt sett vil 'moving average' spore endringene i bakgrunnen. Metoden bruker vekting av ulike piksler i bildet. Dette betyr at hvis bakgrunnen endrer seg raskt vil relativt få piksler ha null vekting, og hvis endringene skjer langsomt vil antallet tidligere piksler med null vekting øke [23]. Algoritmen for 'moving average' er vist under.
Et eksempel på hvordan det vil se ut når bakgrunnen er subtrahert er vist i figur 2.30.

(a) (b)

Figur 2.30: Figuren viser bildet før og etter bakgrunnssubtraksjon.

Form et bakgrunnsestimat $\mathbf{B}^{(0)}$. Ved hver bilderamme F

Oppdatér bakgrunnsestimatet ved å forme:

$\mathbf{B}^{(n+1)} = \dfrac{w_a F + \sum_i w_i B^{(n-i)}}{w_c}$

ved eget valg av w_a, w_i og w_c.

Subtrahér bakgrunnsestimatet fra bilderammen og rapportér verdien til hver piksel hvor størrelsen av forskjellen er større enn en gitt terskel.

slutt

Algoritmen er hentet fra [23]. Vektingen av piksler vil kun bli brukt ved trafikkbildene senere i oppgaven.

Ved enklere bilder vil det ikke være nødvendig å vekte pikslene ulikt grunnet at det er lite støy og små endringer mellom bilderammene. Metoden kan allikevel brukes slik som dette:

$$B_{sub} = \frac{1}{n} \sum_{i=1}^{n} B_{bakgrunn}(i) - B_{objekt} > T$$

I likningen representerer $B_{bakgrunn}$ bakgrunnsbildene, B_{objekt} bildet med objektet en vil subtrahere og B_{sub} det subtraherte bildet. T representerer terskelen for hvor mye pikslene

må ha endret seg ved subtraksjonen. n er antall bakgrunnsbilder som skal brukes til å lage et gjennomsnitt. Det subtraherte bildet vil være et binært bilde som viser de pikslene som har endret seg mest fra bakgrunnsbildet. Disse vil multipliseres med det opprinnelige gråskalabildet og en sitter igjen med kun de mest interessante gråverdiene i bildet.

Kapittel 3

Eksperimentell del

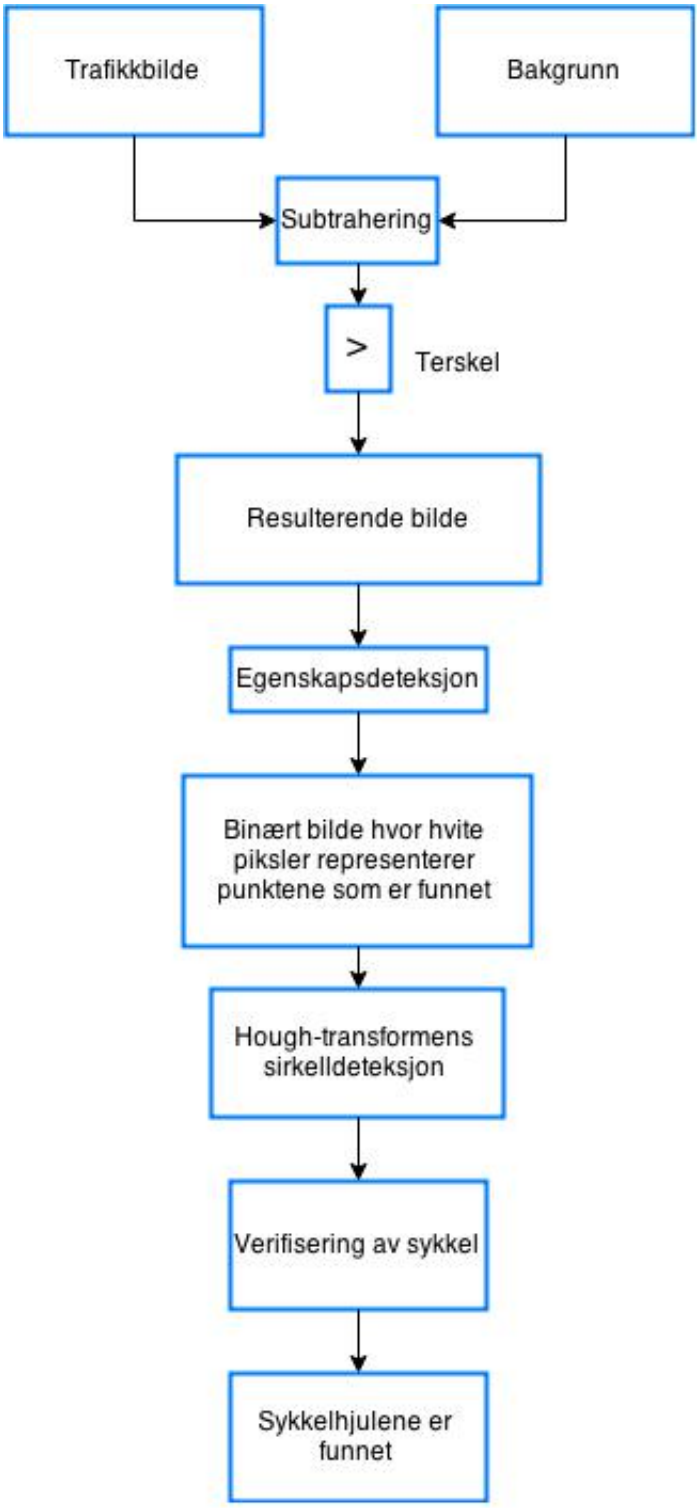
I dette kapittelet vil disse delene av oppgaven bli lagt frem:

- Rammebetingelser for oppgaven
- Verktøy
- Analyse av detektorer
- Optimalisering av valgt detektor

Disse vil inneholde en forklaring av rammebetingelser og verktøy brukt i oppgaven, samt metoden bak det eksperimentelle.

To bildesett vil blir brukt gjennom den eksperimentelle delen av oppgaven, 5.1 og 5.1. Det ene settet vil inneholde enkle bilder i forhold til deteksjonen, det andre vil inneholde bilder tatt ute i trafikken som kunne vært brukt i praksis.

Fremgangsmåten som er valgt for deteksjon av syklene vises i figur 3.1.



Figur 3.1: Fremgangsmåte for deteksjon av sykkelen.

3.1 Rammebetingelser

Det er satt opp noen rammebetingelser for utførelsen av denne oppgaven:

- Hjulene er valgt som objekter i gjenkjenningen av sykkelen.
- Radiusintervall i Hough-transformen er valgt med hensyn på sykkelhjul.
- Bredden av sykkelveien kan ikke være for stor, grunnet radiusintervallet ved Hough-transformen. Hvis radiusintervallet blir for stort vil Hough-transformen bruke lang tid i tillegg til at den ikke vil ha plass til å lagre alle sirklene. Dette intervallet vet man i det kameraet blir satt opp.
- Bildene blir tatt slik at sykkelen sees fra siden. Helst slik at det ikke blir noen vinkel på hjulene. Dette er grunnet Hough-transformen, det vil ikke være mulig å finne hjulene om sirklene er deformerte.
- Sykkelstien er i front av trafikken. Syklisten vil være nærmeste objekt til kameraet.

3.1.1 Svakt perspektiv

Bildene i denne rapporten tilsvarer bildemodellen: Svakt perspektiv. Dette kan en stadfeste ved at kameraet er holdt ved samme posisjon ved alle bildene. Scenen samt objektets dybde er liten i forhold til gjennomsnittsavstanden fra kameraet, dermed kan forstørrelsen sees på som konstant [23]. Med dette trenger en ikke sette opp rotasjons- og translasjonsmatrise.

3.2 Verktøy

I oppgaven er det brukt en bærbar pc med en enkjernet AMD A6-4455M APU 2.10 GHz prosessor. Verktøyet anvendt i oppgaven er i form av Matlab-kode. I dette delkapittelet blir det presentert en kort beskrivelse av hvert enkelt 'skript'/'funksjon.

3.2.1 Åpen kildekode

Koden bak Hough-transformen er hentet fra åpen kildekode. Tre funksjoner er brukt for å kjøre Hough-transformen: circlepoints.m, circle_ houghpeaks.m og circle_ hough.m. Koden er skrevet av David Young og hentet fra exchange files på Matlab sine hjemmesider. [24].

circle_ hough.m

circle_ hough.m setter opp Hough-matrisen, der dimensjonen er avhengig av hvor stort radiusintervallet er satt til. Den beregner alle parameterne for alle mulige sirkler som går gjennom hvert egenskapspunkt, deretter inkrementeres cellene i matrisen som tilsvarer parameterne til de sirklene som får stemmer fra egenskapspunktene.

circle_ houghpeaks.m

circle_ houghpeaks.m finner lokale maksimumspunkt i Hough-matrisen som tilsvarer parameterne til de identifiserte sirklene i bildet. Parameterne blir lagret i en 3xN-matrise. Funksjonen tilbyr også en gaussisk filtrering av Hough-matrisen før maksimumsverdier blir funnet. Dette er for at maksimumsverdier i en matrise påvirket av støy skal være enklere å finne. Etter å ha lokalisert et maksimumspunkt setter funksjonen nabo-verdiene i matrisen til ikke å være maksimumspunkt, 'non-maxima suppression'. Dette er for å unngå å identifisere den allerede identifiserte sirkelen som en ny sirkel.

circlepoints.m

circlepoints.m regner ut heltalls-verdiene for koordinatene som de ulike sirklene skal gå gjennom. Dette må gjøres grunnet at bildeplanet er et diskret rom og ikke et kontinuerlig rom. circlepoints.m blir brukt i circle_ hough.m og når de identifiserte sirklene plottes.

3.2.2 Kode fra MATLAB sine biblioteker

BRISK- og SURF-detektoren brukes ved hjelp av funksjoner i matlab. Disse funksjonene heter: detectBRISKFeatures og detectSURFFeatures.

detectBRISKFeatures:

detectBRISKFeatures returnerer egenskapspunkter funnet ved BRISK-detektoren.

detectSURFFeatures:

detectSURFFeatures returnerer egenskapspunkter funnet ved SURF-detektoren.

3.2.3 Egentilpasset kode

Bakgrunnssub_ enkel.m

Bakgrunnssubtraksjon for de ideelle testbildene. Her med en hvit, urørlig bakgrunn. 'Moving average' uten vekting er grunnlaget bak denne funksjonen.

Bakgrunnssub_ trafikk.m

Bakgrunnssubtraksjon for bildene tatt med trafikk-bakgrunn. Her er metoden 'moving average' implementert med vekting.

testBrisk.m

Et test-skript for BRISK-detektoren. Hough-transformen blir og kjørt i dette skriptet for å godkjenne egenskapspunkter.

testSurf.m

Et test-skript for SURF-detektoren. Hough-transformen blir og kjørt i dette skriptet for å godkjenne egenskapspunkter.

Susan_ kant.m

Egen implementering av SUSAN-detektoren. Laget ut i fra teorien bak detektoren. Hough-transformen blir og kjørt i dette skriptet for å godkjenne egenskapspunkter.

test_ SusanKant.m

Et test-skript for SUSAN-detektoren.

test_ enkleFigurer.m

I analyse-kapittelet testes detektorene på noen enkle figurer, dette skriptet lager figurene som er brukt.

3.3 Analyse av detektorer

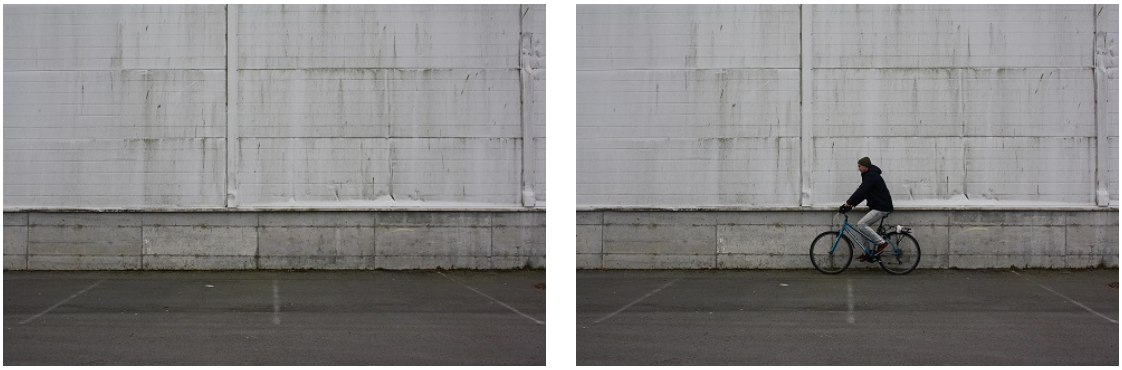
Det vil bli gjort fire ulike tester for å bestemme hvilken detektor som passer best til deteksjon av en sykkel. Detektorene vil bli testet på:

- Enkle figurer som passer formen på objektet som skal detekteres i bildet.
- Et ideelt sykkelbilde.
- Et sykkelbilde ved lysendring.
- Et bilde med okklusjon.

Hvordan de ulike testene skal utføres blir lagt frem i dette kapittelet samt bildebehandlingen som blir gjort i forkant av testene.

3.3.1 Bakgrunnssubtraksjon

Bakgrunnssubtraksjon blir anvendt på både bildesettet med enkel bakgrunn 5.1 og bildesettet med trafikk 5.1. De to ulike metodene beskrevet i teoridelen 2.30 blir anvendt på hvert sitt bildesett. Bildene som blir brukt under testingen er vist i figur 3.2. Ved optimalisering av valgt detektor testes bildene med mer støy. Disse bildene er vist i figur 3.3.



(a) (b)

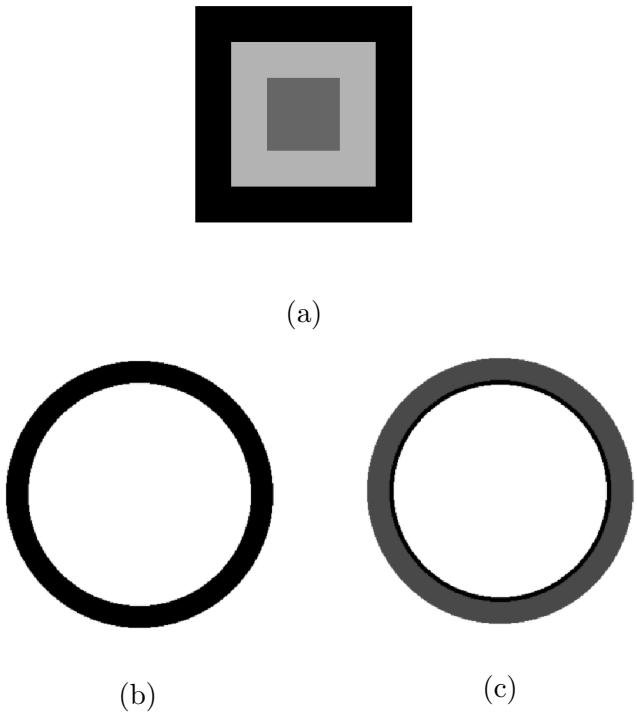
Figur 3.2: Fra venstre: Bakgrunnen og bakgrunnen med objektet som skal segmenteres.

(a) (b)

Figur 3.3: Fra venstre: Bakgrunnen og bakgrunnen med objektene som skal segmenteres

3.3.2 Test 1: Enkle figurer

Her vil en teste detektorene opp mot ulike enkle figurer og vurdere hvilken detektor som opptrer best i forhold til disse. Det er lagt til figurer i kvadratisk og sirkulær form. Det ideelle vil være om detektorene finner egenskapspunkter på sirklene. Sirklene representerer hjulene som skal detekteres i sykkelbildene.
Figurene er laget ved hjelp av Matlab, og det er brukt et gaussisk glattefilter for å lage kantene uskarpe (her med σ lik 0.5).
Detektorene vil være var på gråverdier og for å kunne bruke det samme utgangspunktet som senere i oppgaven er det valgt å normalisere gråverdiene i bildene.
Bildene som er brukt i denne testen vises i figur 3.4. Ved figur 3.4c er det laget to sirkler med ulike gråverdier. Dette er for å teste detektorene ved intensitetsendring. Det samme gjelder figur 3.4a.



(a)

(b) (c)

Figur 3.4: Figuren viser bildene av de enkle figurene brukt i test 1. av analysen av detektorene

3.3.3 Test 2: Ideelt bilde

Detektorene blir testet på et ideelt bilde i forhold til deteksjon av en sykkel. Det som gjør bildet ideelt er:

- Det er valgt ut et bilde med kun syklist som objekt.
- Bakgrunnsstøyen er subtrahert fra bildet.
- Gråverdiene som er igjen etter bakgrunnssubtraheringen er normalisert grunnet lave verdier. Detektorene er avhengig av å kunne utelukke egenskapspunkter gitt en intensitetsendring i bildet, dermed er det viktig at gråverdiene ikke er for lave.

For at deteksjonen av egenskapspunkter skal være god nok er en avhengig av at detektoren finner punkter på objektet som skal gjenkjennes og dette vist ved at Hough-transformen finner sirklene gjennom egenskapspunktene til hjulet. Egenskapspunktene funnet ved detektorene plottes som hvite piksler på en svart bakgrunn, slik at det kun er disse Hough-transformen forholder seg til ved sirkeldeteksjonen.
Bildet som er brukt i denne testen er vist i figur 3.5.

Figur 3.5: Ideelt bilde

3.3.4 Test 3: Lysendring

Et av kriteriene som er satt for at en detektor skal være god er at den er robust mot lysendring. Hvis dette skulle vært realisert i praksis måtte kameraet stått utendørs, dermed vil det være hensiktsmessig å se hvor robuste detektorene er mot lysendring. Det ideelle bildet brukt i test 2 (se figur 3.5) er anvendt i denne testen også. Endringen av lys tilsvarer 0-25 % av det opprinnelige bildets intensitetsverdier. Figurene vil vise endringene 5 %, 15 % og 25 %. I tillegg vil Hough-transformen bli brukt for å vise om punktene som blir funnet kunne passet sirkeldeteksjonen. Det er viktig at det blir funnet nok punkter langs hjulet for at sykkelen skal kunne detekteres.

(a) Det ideelle bildet med 5% lysendring.

(b) Det ideelle bildet med 10% lysendring.

(c) Det ideelle bildet med 25% lysendring.

Figur 3.6: Bildet vist med 5%, 15% og 25% lysendring fra det ideelle bildet.

3.3.5 Test 4: Okklusjon

Definisjonen på okklusjon er at noe blokkerer objektet som skal detekteres. I utgangspunktet betyr dette at objektet havner bak noe. I dette tilfellet vil okklusjonen foregå bak objektet. Det som gjør det til okklusjon er at det gjemmer deler av objektet selv om objektet er nærmest kameraet. Sykkelhjulet vil være vanskeligere å finne grunnet en bil i bakgrunnen. Bilen vil dekke en del av hjulets form.

Det er viktig at detektoren kan finne syklisten tross andre objekter i bakgrunnen. Ved bruk i trafikken vil det være flere objekter i bildet. Hough-transformen blir også anvendt i denne testen for å illustrere beste detektor. Bildene som brukes i denne testen er vist i figur 3.7 og hentet fra 5.1.

Figur 3.7: Første bildet som brukes til å teste detektorene ved okkusjon. Hentet fra 5.1.

Figur 3.8: Andre bildet som brukes til å teste detektorene ved okklusjon. Hentet fra 5.1.

3.3.6 Valg av detektor

Resultatene fra testene vil bli diskutert og detektor som passer oppgaven best blir valgt.

3.4 Optimalisering av valgt detektor

Optimaliseringen av den utvalgte detektoren vil foregå i form av testing på 5.1 og 5.1. For at detektoren skal kunne fungere opp mot trafikkbilder må den prestere ved et enklere bilde. Optimaliseringen vil bestå av å gjøre endringer slik at:

- Egenskapspunkter som ikke tilhører sykkelen er fjernet. Punkter som blir funnet utenfor sykkelen vil kun oppfattes som støy.
- Tiden egenskapsdetektoren bruker på deteksjonen er redusert.
- Detektoren er robust mot støy som kan forekomme.
- Detektoren er tilpasset sykkeltelling.

3.4.1 Tilpasning for sykkeltelling

Gitt sirklene Hough-transformen finner som de to mest fremtredende er det usikkert om disse tilhører sykkelen. Derfor er det implementert kode som kontrollerer (ut i fra noen kriterier) om sykkel er funnet. Kriteriene er som følger:

- Avstand mellom hjulene (avstand mellom sentrene av sirklene som er funnet, x-retning) må være en gitt størrelse.
- Vinkel mellom hjulene (senterpunktene til sirklene i y-retning vil være tilnærmet like). Hjulene vil ligge på en horisontal linje, dermed vil vinkelen mellom de være liten.

Hough-transformen legger sirklene som er funnet i en rekkefølge etter hvor mange punkter i bildet som treffer sirkelen. Dermed er det større sannsynlighet for at Hough-transformens

første sirkel er et av hjulene. Dette tas og med i betraktning.

3.4.2 Test ved trafikkbilder

Den optimaliserte detektoren testes mot bildesettet med trafikkbilder 5.1. Her vil det være trafikk i bakgrunnen og flere objekter til stede. Dette gjør det vanskeligere ved deteksjon. Bildebehandling vil være brukt i form av bakgrunnssubtraksjon. Et av bildene fra bildesettet som brukes vises i figur 3.9. Hough-transformens sirkeldeteksjon brukes i dette tilfellet også.

Figur 3.9: Trafikkbilde fra 5.1

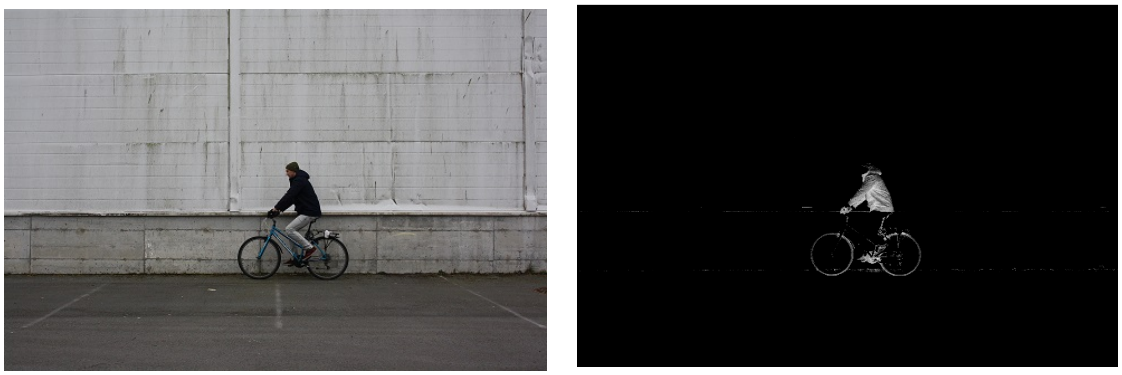
Kapittel 4

Resultater og diskusjon

I dette kapittelet fremlegges resultatene fra testingen og diskusjonen rundt disse. Det er blitt gjort forarbeid på bildene før testene, resultatet av det vil også bli gjort rede for i dette kapittelet.

4.1 Resultat av bakgrunnssubtraksjonen

Resultatet av bakgrunnssubtraksjonen for bildet brukt i analysen er vist i figur 4.1.



(a) Ideelt bilde

(b) Bakgrunn subtrahert fra ideelt bilde

Figur 4.1: Det ideelle bildet med subtrahert bakgrunn

Bakgrunnssubtraksjon er i tillegg anvendt på bildesett 5.1. Her brukes 'moving average' med vekting av piksler. Ved estimatet av bakgrunnen som trekkes fra er det lagt til verdier

for w_a, w_i og w_c.

$w_a = 0.3$

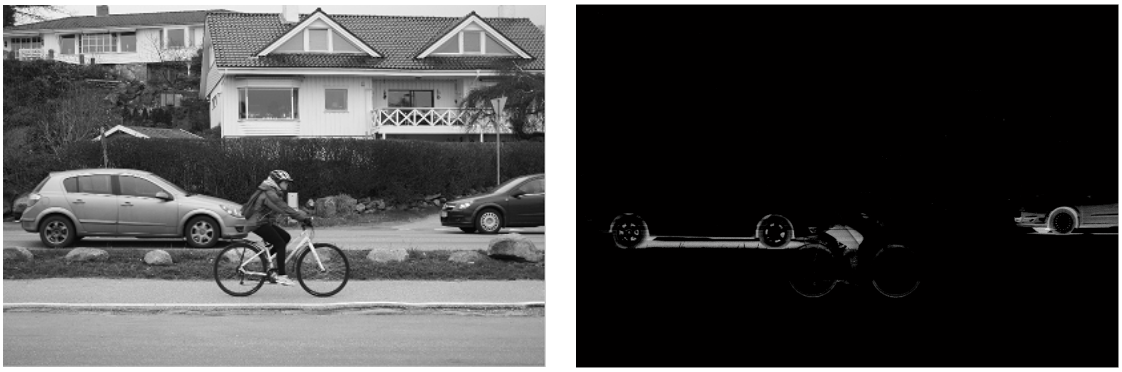
$w_i = 0.2$

$w_c = 1.2$

Vektingen vises i likning (4.1.1). $B^{(n+1)}$ representerer bakgrunnsestimatet og $B^{(n-i)}$ forrige bakgrunnsestimat.

$$B^{(n+1)} = \frac{0.3F + \sum_i 0.2B^{(n-i)}}{1.2} \tag{4.1.1}$$

Bakgrunnsestimatet som blir beregnet ved likning (4.1.1) blir subtrahert fra bildet med objektet som skal segmenteres. Verdiene ved vektingen er valgt ut etter testing av ulike trafikkbilder fra 5.1. Verdiene som gir best segmentering av objektene vil tilsvare beste vektingen av bakgrunnsestimatet. Når subtraksjonen er fullført settes det en terskel som kun tillater verdier som har stor forskjell mellom det opprinnelige bilder i gråtoner og bakgrunnsestimatet. Med dette plukkes pikslene som har endret seg mest ut og segmenteringen er fullført. Tersklingen er satt til 8 i disse bildene. Intensitetene som har forandret seg mindre enn dette vil bli satt til 0 og gråverdiene til de resterende blir hentet fra gråskalabildet. Resultatet av bakgrunnssubtraksjonen er vist i figur 4.2.



(a) Trafikkbilde uten bakgrunnssubtraksjon

(b) Trafikkbilde med subtrahert bakgrunn

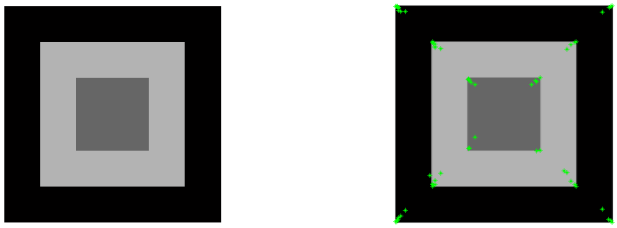
Figur 4.2: Et trafikkbilde med og uten subtrahert bakgrunn.

4.2 Sammenligning av BRISK, SURF og SUSAN

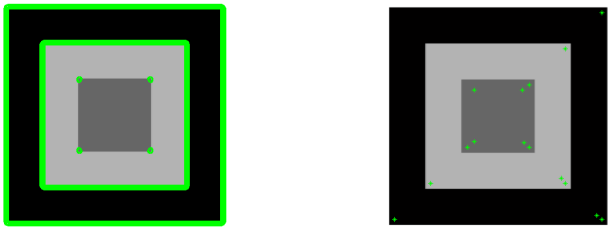
Resultatene av de ulike testene ved detektorene blir fremlagt i dette delkapittelet. Grunnet matlab sine funksjoner av SURF- og BRISK-detektor er det ikke gjort noen endringer med disse. SUSAN-detektoren som er en egenimplementert kode er satt med gråverditerskel lik 57 og geometrisk terskel lik $0.625 \cdot n_{maks}$. Den høye gråverditerskelen er satt slik grunnet normalisering av gråverdiene ved de enkle bildene.

4.2.1 Detektorene testet på enkle figurer

Resultatet av detektorene testet på de enkle figurene er vist i 4.3. Grønne punkter i figuren representer egenskapspunkter funnet ved deteksjonen.



(a) Det opprinnelige bildet (b) BRISK-detektoren

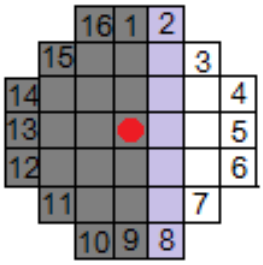


(c) SUSAN-detektoren (d) SURF-detektoren

Figur 4.3: Test av detektorene på firkanter med ulik intensitet.

BRISK-detektoren finner punkter med egenskapen: hjørne. Dette ser man godt i første figur hvor alle grønne punkter er plottet ved hjørnene i kvadratene. Ytterste sirkelen til BRISK-detektoren er avhengig av å ha 9 av 16 piksler tilstrekkelig mørkere eller

lysere enn senter-pikselen og det er dette som gjør at kantene ikke blir plukket ut som egenskapspunkter. Se figur 4.4.



Figur 4.4: BRISK-detektorens maske over en kant av kvadratet. Rødt punkt representerer senteret av masken.

SUSAN-detektoren har en terskel hvor hjørner og kanter blir tatt med som egenskapspunkter. Ved den innerste firkanten ser en at ulikheten mellom intensitetene blir for liten og at kantene ikke blir detektert. Dette er grunnet terskelen T (gråverditerskelen) som sier noe om hvor stor ulikheten i intensitet kan være mellom pikslene for at det skal oppfattes som et egenskapspunkt. SURF-detektoren finner få punkter, men de som er detektert er ved hjørnene av kvadratet. Dette viser at denne detektoren finner punkter med egenskapen hjørne.

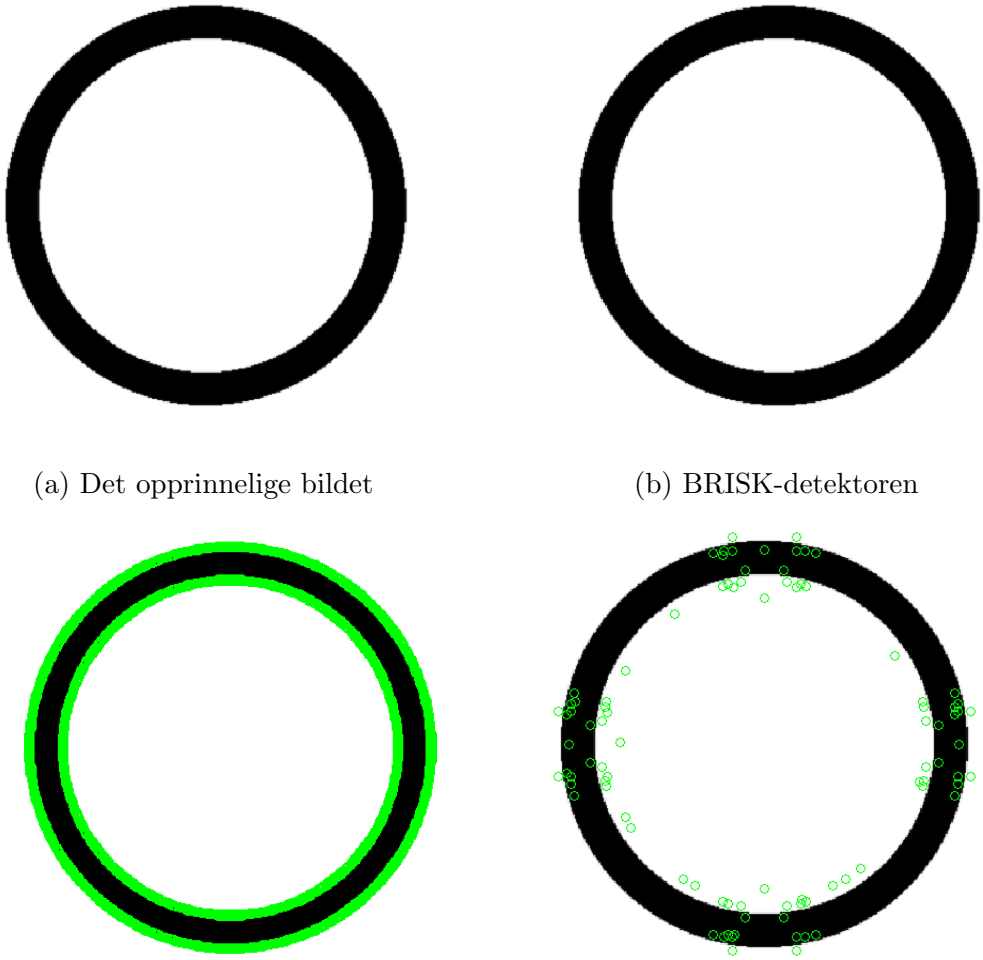
En tabell som viser antall punkter funnet, tiden brukt på deteksjon og tid brukt per punkt er vist i 4.1.

Detektor	Tid[s]	Antall egenskapspunkter	Tid per egenskapspunkt[s]
BRISK	0.18898	63	0.003
SUSAN	3.81945	3988	0.00095774
SURF	0.087498	15	0.0058

Tabell 4.1: Antall egenskapspunkter funnet ved BRISK-, SUSAN- og SURF-detektoren samt tidsbruk.

En kan ut i fra tabellen se at SUSAN-detektoren finner flest egenskapspunkter i bildet. I

tillegg bruker detektoren kortest tid per punkt.



(a) Det opprinnelige bildet

(b) BRISK-detektoren

(c) SUSAN-detektoren

(d) SURF-detektoren

Figur 4.5: Figur med sirkulær form, testet på BRISK-, SUSAN- og SURF-detektoren. Grønne punkter representerer egenskapspunkter funnet ved deteksjon.

Ved 4.5 finner ikke BRISK-detektoren noen egenskapspunkter ved sirkelen. Dette er av samme grunn som ved kvadratets kanter. Det er ikke nok piksler som vil være tilstrekkelig mørkere eller lysere i intensitet i forhold til senter-pikselen. SUSAN-detektoren finner kantene både på innsiden og yttersiden av sirkelen. Intensitetsforskjellen er stor og dette gjør det enkelt for detektoren å finne kanten. I tillegg er SUSAN-detektorens geometriske terskel satt til 0.625, som betyr at detektoren slipper igjennom det meste av kantene. SURF-detektoren finner flest punkter i ren x- og y-retning. I følge artikkelen som om-

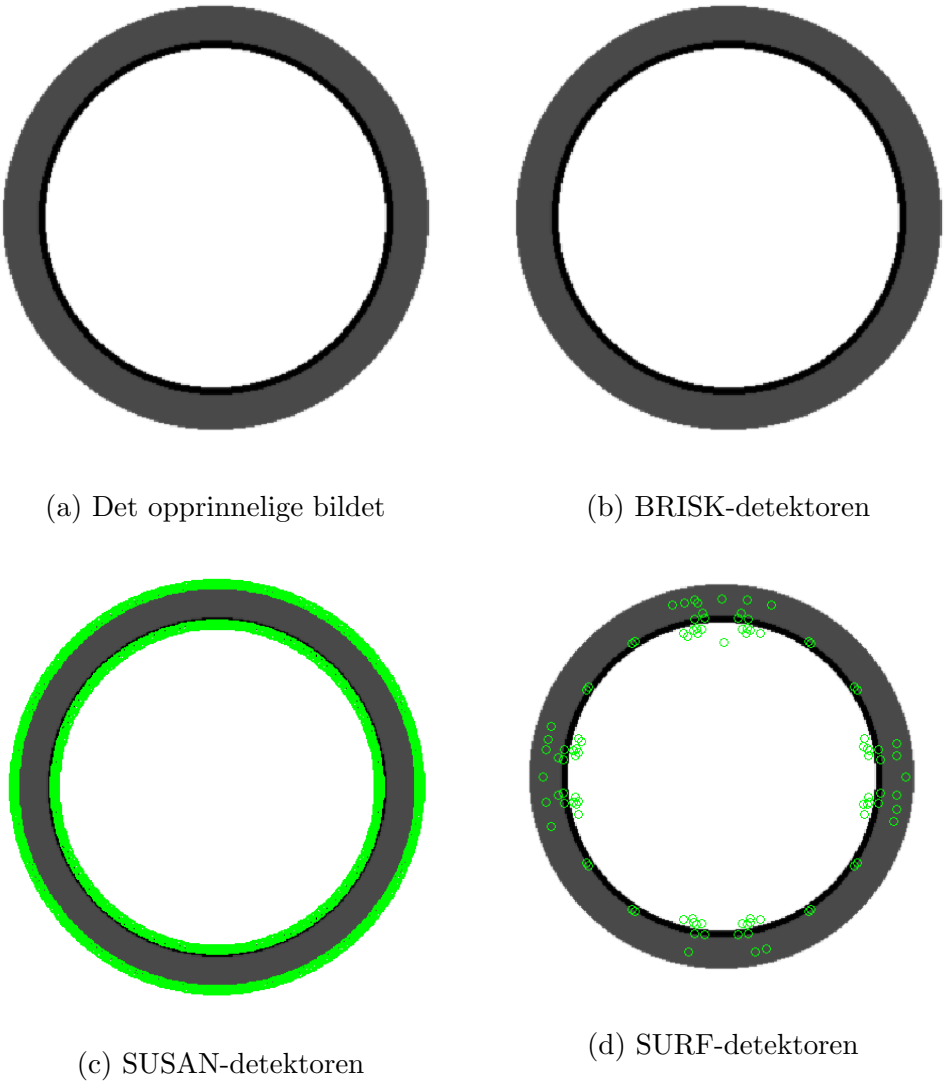
handler SURF [5] kommer dette av at Gaussiske filtre er optimale ved ulike skalaer, men i praksis må de diskretiseres og kuttes. Dette fører til at repeterbarheten under bilderotasjoner rundt oddetalls multiplisering av $\frac{\pi}{4}$ reduseres. Dette er en svakhet alle Hessian-baserte detektorer har generelt. Repeterbarheten oppnår maksimum rundt $\frac{\pi}{2}$ og det kommer av den kvadratiske formen på filteret anvendt under SURF-deteksjon [5]. Dette kommer veldig godt frem i bildet grunnet at alle vinklene er synlige. Rundt vinkelen $\frac{\pi}{2}$ er det mest egenskapspunkter å finne. Det er også tydelig i figur 4.5d at oddetalls multiplisering av $\frac{\pi}{4}$ er detektorens svakhet, hvor en kan se at det ikke er funnet punkter.

En tabell som viser antall punkter, tid brukt under deteksjon og tid per egenskapspunkt er vist i figur 4.2.

Tabell 4.2: Tidsbruk og antall egenskapspunkter funnet ved BRISK-, SUSAN- og SURF-detektoren på sirkelfiguren.

Detektor	Tid[s]	Antall egenskapspunkter	Tid per egenskapspunkt[s]
BRISK	0.0292858	0	0
SUSAN	3.796568	2160	0.0018
SURF	0.090275	94	0.00096037

I tabellen ser en at SUSAN-detektoren finner flest punkter ved figuren, men i forhold til per punkt er SURF-detektoren mer effektiv.



(a) Det opprinnelige bildet

(b) BRISK-detektoren

(c) SUSAN-detektoren

(d) SURF-detektoren

Figur 4.6: Sirkulær figur med to ulike gråverdier.

Ved figur 4.6 opptrer egenskapsdetektorene likt som ved figur 4.5. Det samme mønsteret vises ved SURF-detektoren.
Den innerste sirkelen er lagt til grunnet intensitetsforskjeller og test av dette. Sykkelhjulet vil ha noe lignende grunnet skygge. Hvis en forstørrer figur 4.6c ser en at SUSAN-detektoren kun finner kanten på innsiden av sirkelen og at intensitetsforskjellen ikke er stor nok mellom den innerste (svart) og ytterste sirkelen (grå).

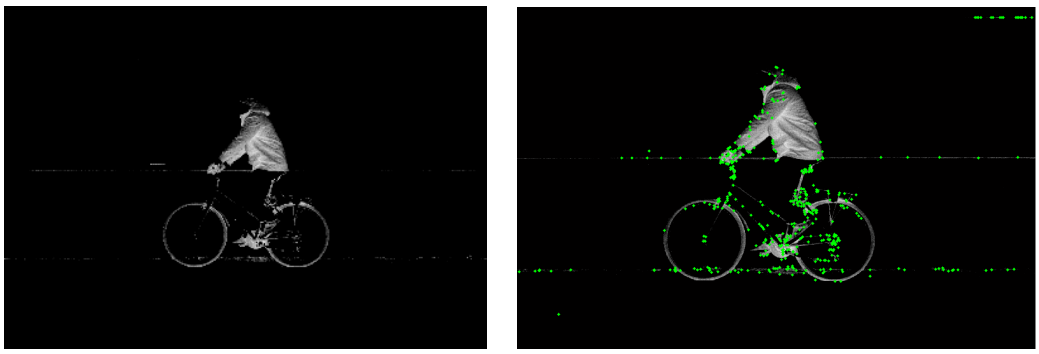
En tabell som viser antall punkter funnet, tiden brukt under deteksjonen og tid per egenskapspunkt er vist i 4.4. Ved denne figuren utfører SURF- og SUSAN-detektoren ved

ulik tid, men per egenskapspunkt bruker de lik tid.

Detektor	Tid[s]	Antall egenskapspunkter	Tid per egenskapspunkt[s]
BRISK	0.031690	0	0
SUSAN	3.992101	2104	0.0019
SURF	0.210107	101	0.0021

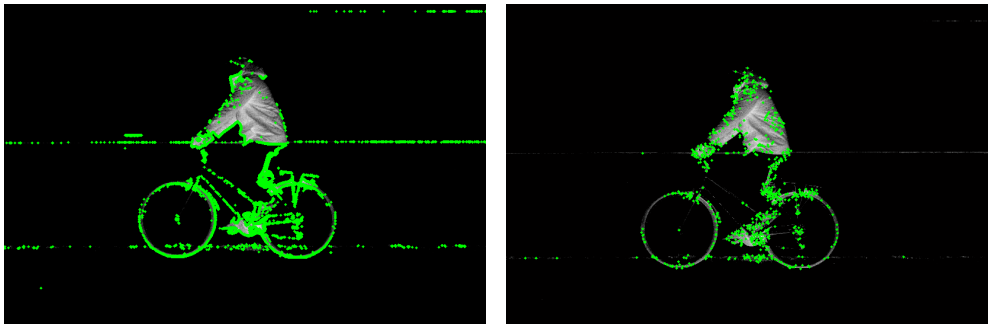
Tabell 4.3: I denne tabellen vises tidsbruk og antall egenskapspunkt funnet ved deteksjonen.

4.2.2 Detektorene testet på et ideelt sykkelbilde



(a) Det opprinnelige bildet med subtrahert bakgrunn

(b) BRISK-detektoren



(c) SUSAN-detektoren

(d) SURF-detektoren

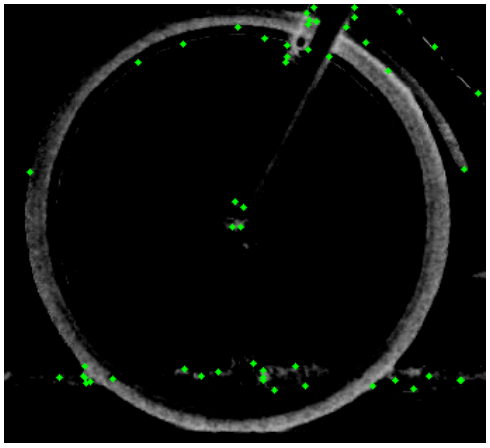
Figur 4.7: Egenskapspunktene funnet på det ideelle bildet ved hjelp av BRISK-, SUSAN-, og SURF-detektoren. De grønne punktene representerer egenskapspunktene som er funnet.

Figur 4.7 er tatt med for å vise hvilke andre punkter i bildet utenom objektet som blir funnet.
SUSAN-detektoren finner punkter ved hjulet, men punkter utenfor sykkelen detekteres i tillegg. Disse vil være unødvendige i denne oppgaven og blir dermed regnet som støy. At detektoren finner punkter utenom de relevante vil kun gjøre deteksjonen tregere. Både i forhold til deteksjonen av punkter og i forhold til sirkeldeteksjonen.
SURF er detektoren med færrest punkter utenom objektet, dette blir sett på som positivt.
BRISK-detektoren finner punkter i bildet, men disse punktene ligger ikke på sykkelens hjul. Dermed vil det bli vanskelig å finne hjulene ved sirkeldeteksjonen.
Antall punkter funnet ved deteksjonen og tidsbruket er vist i tabell 4.4. Denne viser at SUSAN- og SURF-detektoren bruker like lang tid per egenskapspunkt selv om SUSAN-detektoren finner flest punkter. BRISK-detektoren er den desidert raskeste av de tre. Det ble presentert ved teoridelen at en av BRISK-detektorens fordeler er at den er rask og det vises her.

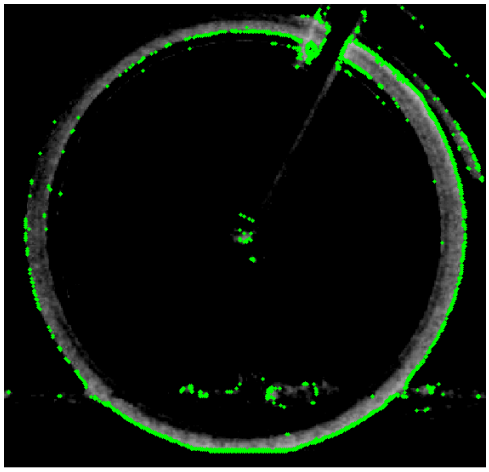
Detektor	Antall egenskapspunkter	Tid[s]	Tid per egenskapspunkt[s]
BRISK	533	0.433127	0.00081262
SUSAN	9828	181.319759	0.0181
SURF	752	3.975720	0.0053

Tabell 4.4: Tabellen viser antall egenskapspunkt funnet ved de ulike detektorene og tid brukt under deteksjonen.

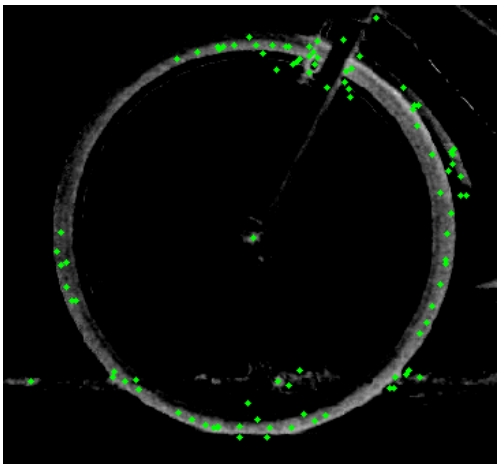
Et bilde som viser sykkelens hjul ved deteksjon er vist i figur **??**. Her ser en tydelig egenskapspunktene (vist med grønt) funnet ved detektorene.



(a) BRISK-detektoren



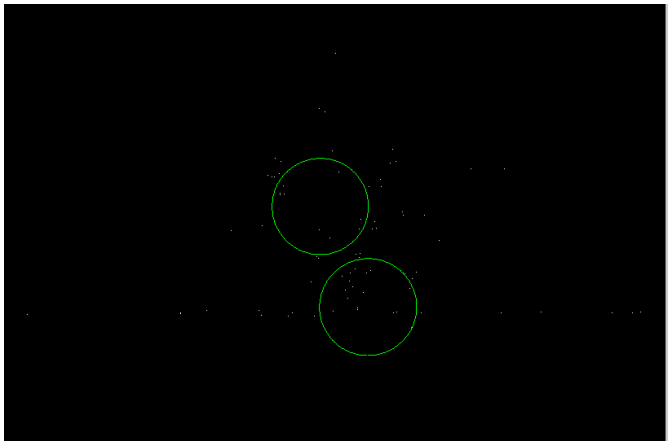
(b) SUSAN-detektoren



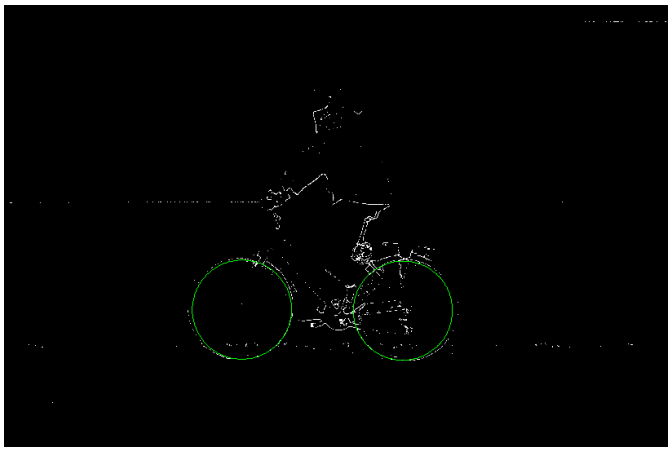
(c) SURF-detektoren

Figur 4.8: Deteksjon av egenskapspunkter. Et grønt punkt representerer et egenskapspunkt.

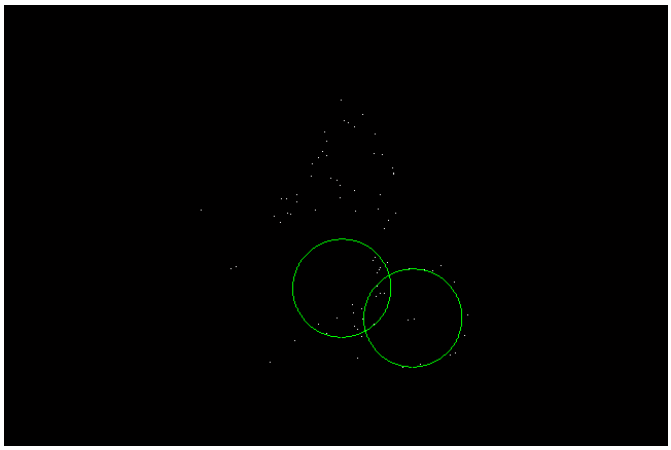
En ser ved figur **??** at egenskapspunktene funnet ved SURF-detektoren danner samme mønster som ved testingen av detektorene på enkle figurer 4.7d. En er avhengig å finne nok egenskapspunkter som karakteriserer sykkelhjulet for at dette skal kunne bli brukt videre i deteksjonen. Derfor er punktene fra testen 'tegnet' som hvite piksler på en svart bakgrunn og deretter er Hough transformen testet på disse punktene. Hjulets radius på dette bildet er funnet ved hjelp av 'imtool' i forkant av testingen.



(a) BRISK-detektoren



(b) SUSAN-detektoren



(c) SURF-detektoren

Figur 4.9: De hvite pikslene representerer egenskapspunkter funnet ved deteksjonen til de ulike detektorene. De grønne sirklene er laget som et resultat av Hough-transformen anvendt på disse punktene.

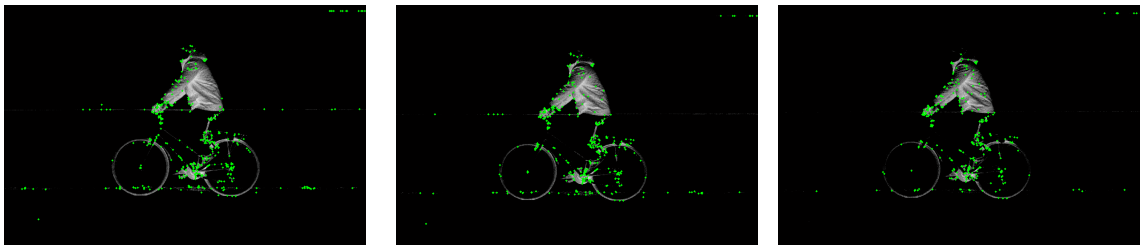
Om man forstørrer bildet til BRISK-detektoren i figur 4.9, vil en bare se en haug med punkter. Det vil ikke se ut som syklisten fra det opprinnelige bildet. Hjulene virker ikke fremtredende i forhold til andre punkter, dermed finner ikke Hough-transformen hjulene ved sirkeldeteksjonen.

Det er kun SUSAN-detektoren som finner nok punkter på hjulet til at Hough-transformen finner sykkelen. Ved SURF-detektoren blir de fleste punktene funnet ved syklisten og ikke selve sykkelen. Dette er grunnen til at Hough-transformen finner en av de mest fremtredende punktene ved syklisten. Grunnen til at SURF-detektoren gjør dette er at den er spesialisert til å finne 'blob'-egenskapspunkter, områder med flere piksler av samme intensitet.

4.2.3 Lysendring

BRISK-detektoren ved lysendring

Figur 4.10 viser at jo mer lysendring, desto flere punkter som ikke er relevant for sirkeldeteksjonen fjernes.



(a) 5% lysere (b) 15% lysere (c) 25% lysere

Figur 4.10: BRISK-detektoren ved 5 %, 15 % og 25 % lysendring. Grønne punkter representerer egenskapspunkter.

BRISK-detektorens egenskapspunkter er testet med Hough-transformens sirkeldeteksjon. Resultatet er vist i figur 4.11.

(a) 5% lysere (b) 15% lysere (c) 25% lysere

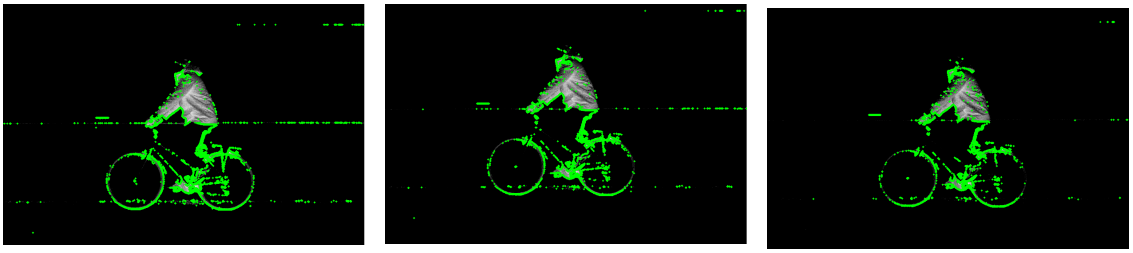
Figur 4.11: BRISK-detektoren ved 5%, 15% og 25 % lysendring. De røde sirklene representerer hjulene funnet ved Hough-transformen.

Detektoren har en redusering på 159 punkter etter 25% endring av lyset. Punktene som er redusert er utenfor objektet, men detektoren finner få punkter på sykkelen i utgangspunktet og ingen av hjulene blir funnet ved sirkeldeteksjonen.

Se 5.2 for å se tidsbruk og antall punkter detektert under lysendringen.

SUSAN-detektoren ved lysendring

Figur 4.12 viser tydelig at antall punkter funnet ved egenskapsdeteksjonen reduseres når lysendringen øker. Fordelen er at det er punkter som ligger utenfor sykkelen som forsvinner.



(a) 5% lysere (b) 15 % lysere (c) 25% lysere.

Figur 4.12: SUSAN-detektoren ved 5%, 15% og 25 % lysendring. Grønne punkter representerer egenskapspunkter funnet ved deteksjonen.

Hough-transformens sirkeldeteksjon er testet på punktene funnet ved lysendringen. Resultatet av dette er vist i figur 4.13.

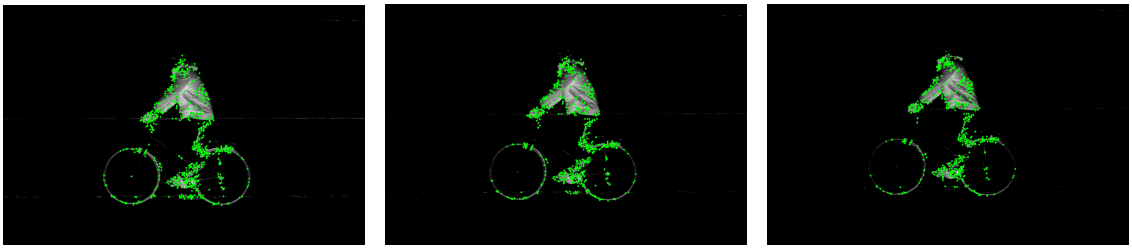
(a) 5% lysere (b) 15 % lysere (c) 25% lysere

Figur 4.13: SUSAN-detektoren ved 5%, 15% og 25 % lysendring. De røde sirklene representerer hjulene funnet ved Hough-transformen.

Selv med en redusering på 4202 punkter blir hjulene funnet ved sirkeldeteksjonen til Hough-transformen. Se tabell 5.3 for å se tidsforbruket og antall punkter detektert ved lysendringen.

SURF-detektoren ved lysendring

SURF-detektoren finner få punkter utenfor objektet før lysendring og den finner ikke flere med. De mest fremtredende punktene funnet ved SURF-detektoren er på syklisten. Fra figur 4.14 ser en at en del av punktene knyttet til hjulene blir borte ved lysendringen.



(a) 5% lysere (b) 15 % lysere (c) 25% lysere

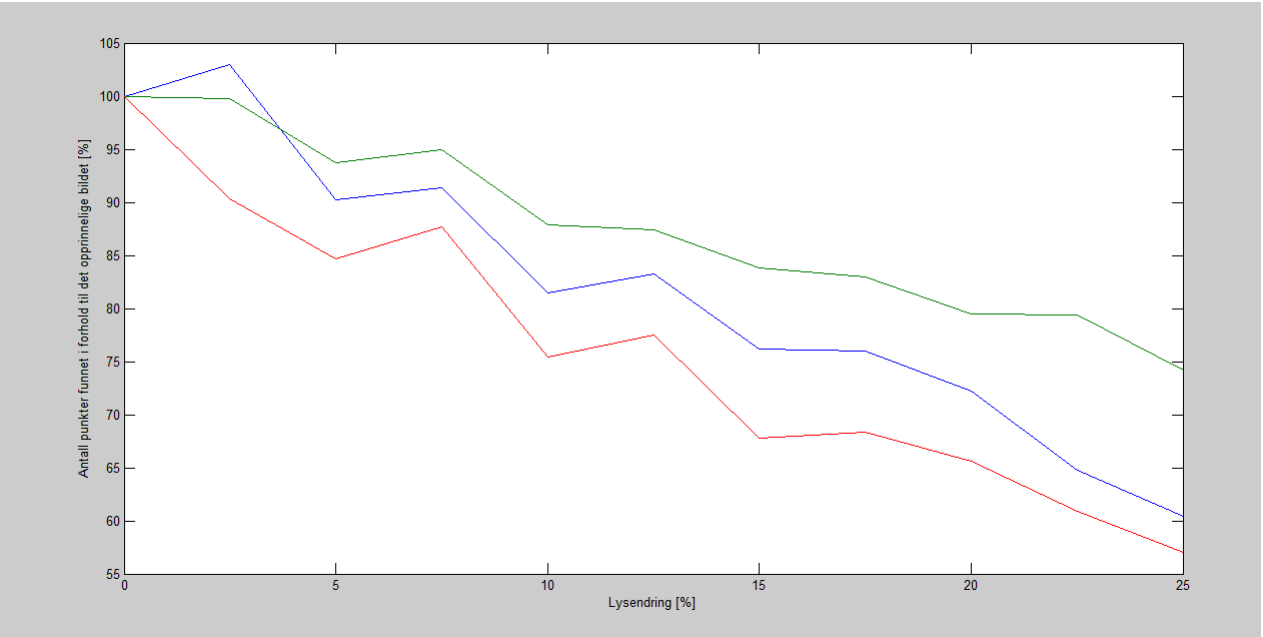
Figur 4.14: SURF-detektoren ved 5%, 15% og 25% lysendring. Grønne punkter representerer egenskapspunkter.

(a) 5% lysere (b) 15 % lysere (c) 25% lysere

Figur 4.15: SURF-detektoren ved 5%, 15% og 25 % lysendring. De røde sirklene representerer hjulene funnet ved Hough-transformen.

Se tabell 5.1 for å se antall punkter funnet og tidsbruk ved lysendring av bildet.

For å kunne måle detektorene opp mot hverandre er prosentandelen fra det opprinnelige bildet til lysendring beregnet og plottet i figur 4.16. Begge aksene er satt oppi henhold til prosent av opprinnelig bilde.



Figur 4.16: Egenskapsdetektorene og antall punkter funnet ved lysendring. Blå: BRISK, grønn: SURF, rød: SUSAN.

Ut i fra grafen vist i 4.16 ser en at SURF-detektoren er mest robust i forhold til lysendring. SUSAN finner flest antall punkter, men er og den detektoren som mister flest i forhold til opprinnelig bilde.
Som vist i figuren 4.15 finner ikke SURF-detektoren hjulene som tilhører sykkelen tross robustheten. Tallene brukt i grafen er hentet fra 5.1, 5.2 og 5.3.

4.2.4 Okklusjon

I figur 4.17 vises de to mest fremtredende sirklene funnet av Hough-transformens sirkeldeteksjon gitt egenskapspunktene til detektorene.

(a) BRISK-detektoren

(b) SUSAN-detektoren

(c) SURF-detektoren

Figur 4.17: Sirklene funnet ved Hough-transformen plottet på det bakgrunnssubtraherte bildet.

Her er sykkelen litt nærmere kamera enn bilen og en kan se mesteparten av hjulet på undersiden, dette gjør det enkelt å finne hjulene.

Antall egenskapspunkter funnet samt tidsbruk ved deteksjonen er vist i tabell 4.5.

Detektor	Antall egenskapspunkter	Tid[s]	Tid per egenskapspunkt[s]
BRISK	801	0.450202	0.000562
SUSAN	23 771	192.824290	0.0081117
SURF	2693	4.147390	0.00154

Tabell 4.5: Oversikt over deteksjon ved okklusjon 1.

Igjen er BRISK-detektoren den raskeste detektoren. SUSAN- og SURF-detektoren finner egenskapspunktene som Hough-transformen trenger for sirkeldeteksjon av hjulene, men SURF-detektoren bruker færre punkt og er raskere.
Test av okklusjons-bilde 2 er vist i 4.18.

(a) BRISK-detektoren

(b) SUSAN-detektoren

(c) SURF-detektoren

Figur 4.18

I dette bildet er sykkelen gjemt mer av bilen enn på forrige bilde. SUSAN-detektoren finner fortsatt nok punkter på hjulet til at sirkeldeteksjonen fungerer. SURF-detektoren finner flere punkter på bilen og mister det ene hjulet ved sirkeldeteksjonen. Dette kommer av at SURF-detektoren finner større område ved bilen som er sammenhengende enn på sykkelen. BRISK-detektoren finner i dette tilfellet flere punkter på syklisten enn på sykkelen.

Detektor	Antall egenskapspunkter	Tid[s]	Tid per egenskapspunkt[s]
BRISK	909	1.768897	0.0019
SUSAN	27258	197.669598	0.0073
SURF	2686	3.811218	0.00142

Tabell 4.6: Oversikt over deteksjon ved okklusjon 2.

Antall punkter og tidsforbruket til de ulike detektorene er vist i tabell 4.6.

4.3 Valg av detektor

- BRISK-detektoren: Deteksjonen av egenskapspunkter utføres raskt, men punktene som blir funnet tilhører ikke sykkelhjulet. Punktene detektoren er ute etter er for skarpe til å passe hjulets form. Detektoren kunne ikke blitt brukt i denne sammenheng.
- SUSAN-detektoren: Detektoren opptrer robust ved de enkle figurene, det ideelle bildet og okklusjon. Hough-transformen finner sykkelhjulene ved alle testene. Ulempene med denne detektoren er at mange punkter blir fjernet ved lyseendring og at det detekteres mange støypunkter i forhold til objektgjenkjenningen.
- SURF-detektoren: Detektoren klarer å skille ut mange av punktene som vil bli tolket som støy i objektgjenkjenningen. Det detekteres punkter på hjulene, men største delen av egenskapspunktene havner på syklisten. Hough-transformen vil dermed opptre ustabilt i forhold til sirkeldeteksjonen av hjulet. Det vil være større sannsynlighet for å finne sirkler på syklisten enn hjulene grunnet 'blob'-egenskapene ved syklisten.
- I forhold til tidsbruk per egenskapspunkt er BRISK-detektoren den desidert raskeste. SURF- og SUSAN-detektoren ligger likt i forhold til tidsbruk.

SUSAN-detektoren blir valgt til denne oppgaven grunnet at den er enkel justere i forhold til de negative sidene detektoren har. BRISK og SURF finner ikke nok punkter på det ønskede objektet til at en kan bruke disse på realistiske bilder (trafikkbilder). Man ser ut i fra testene at det vil være en fordel å lete etter egenskapspunkter som ligner kanter. Dette er lett å endre på med SUSAN-detektoren hvor en selv kan sette inn verdi for den geometriske terskelen. Redusering av tidsbruket til SUSAN-detektoren vil være mulig å redusere ved å endre parameterne i detektoren. Dette gjør metoden justerbar.

4.4 Optimalisering av SUSAN-detektoren

Ved optimalisering av SUSAN-detektoren vil det fokuseres på å redusere egenskapspunkter som blir funnet utenfor sykkelen. Redusering av detektorens tidsbruk ved deteksjon er og en viktig del av optimaliseringen av SUSAN-detektoren. Parameterne som endres på for å oppnå dette:

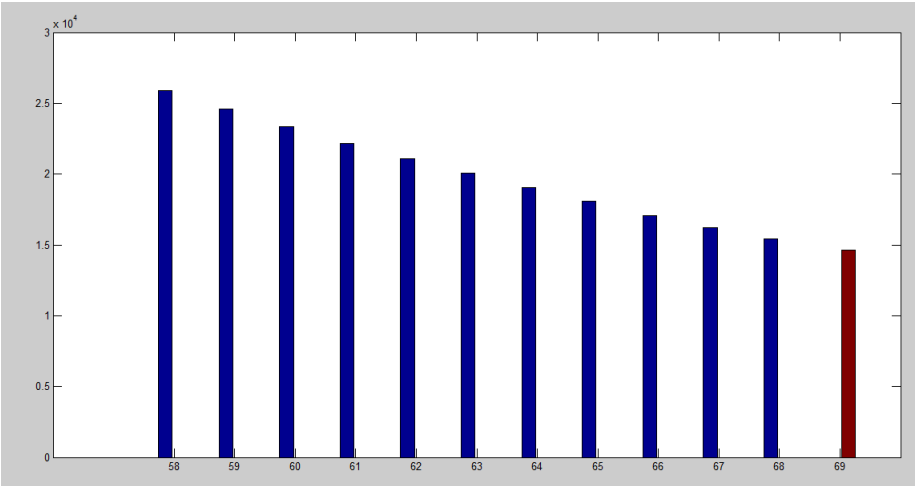
- Geometrisk terskel
- Gråverditerskel
- Maskestørrelse

Først vil en finne en kombinasjon av geometrisk terskel og gråverditerskel som passer egenskapspunktene til hjulet. Dette gjøres ved å teste begge parameterne hver for seg, deretter sammen.

Som 'godkjenning' av terskelen brukes Hough-transformen. Transformen vil settes til å lete etter de to mest fremtredende sirklene, hvis ikke disse blir funnet vil detektoren ha fjernet de avgjørende punktene ved sirklene. Dermed er tersklene for høye. Bildene som testes vil i første omgang være med enkel bakgrunn. Hvis ikke detektoren fungerer optimalt på disse, vil den heller ikke fungere ved trafikkbildene som er mer realistiske.

Ved testing av hver enkelt terskel brukes bildet 'Sykkel7' fra 5.1. Ved kombinasjon av tersklene testes bildene: 'Sykkel6' - 'Sykkel10' fra 5.1.
Geometrisk terskel settes til $0.625 \cdot n_{maks}$ og gråverditerskel til 57, slik som under sammenligningen mot de tidligere detektorene.

Figur 4.19 viser endringene i antall punkter gitt ulike gråverditerskler. Den røde søylen representerer terskelen hvor sykkelhjulene ikke blir funnet.

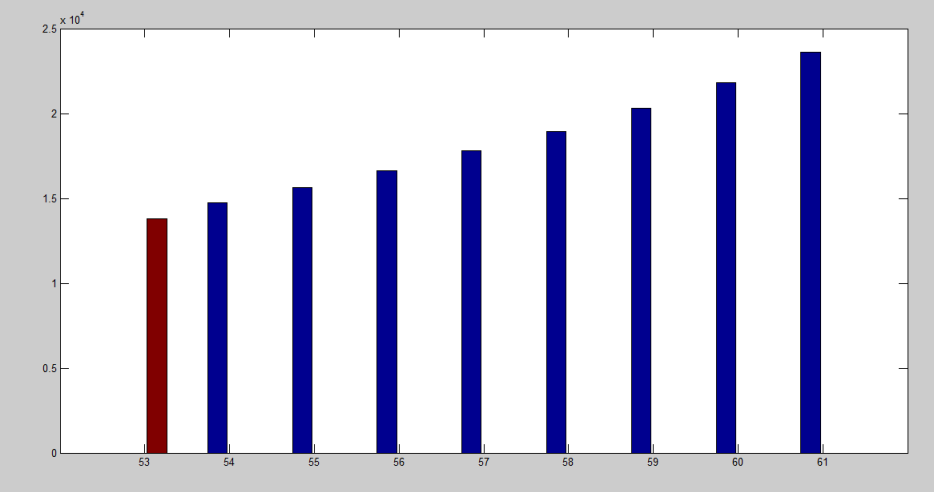


Figur 4.19: Antall egenskapspunkter funnet av SUSAN-detektoren ved ulike gråverditerskler. X-aksen representerer gråverdiene og y- aksen antall punkter. Den røde søylen vil representere terskelen hvor Hough-transformen ikke lenger finner hjulene.

Gråverditerskel 68 gir en redusering på 10506 punkter fra terskel 58 5.4. Og Hough-transformen finner sykkelhjulene tross at punkter er fjernet. Dette vil si at punktene som er fjernet er støypunkter som ikke trengs til objektgjenkjenningen.

Gråverditerskelen vil være avhengig av gråverdiene som allerede finnes i bildet. Bakgrunnssubtraksjonen vil dermed ha stor innvirkning på terskelen. Til akkurat dette bildesettet vil gråverdien kunne presses til 68.

Deretter presses geometrisk terskel ned. Jo mindre den er, jo mindre er formen på egenskapspunktene som blir funnet. En antagelse gjort er at sykkelhjulets form vil ligne en kant mer enn et hjørne. Endringene er vist i figur 4.20 hvor den røde søylen representerer terskelen hvor sykkelhjulene ikke blir funnet.



Figur 4.20: Antall egenskapspunkter funnet av SUSAN-detektoren ved endring av den geometriske terskelen. X-aksen representerer terskelverdiene og y- aksen antall punkter. Den røde søylen representerer terskelen hvor detektoren ikke finner punkter nok til Hough-transformen.

Hough-transformen sliter med å finne sykkelhjulene ved samme antall egenskapspunkter funnet som ved endring av gråverditerskel.

Når tersklene skal kombineres er det viktig at de rette punktene blir detektert. Egenskapspunktene som hører til hjulet vil ligne en buet kant, dermed vil den geometriske terskelen være mindre enn en kant, men større enn et hjørne. Antall punkter funnet under testene på de enkle bildene er mange, dermed vil en kunne presse den geometriske terskelen ned. Gråverditerskelen holdes til 57.

GeoT = 0.61·n_{maks} og T = 57:

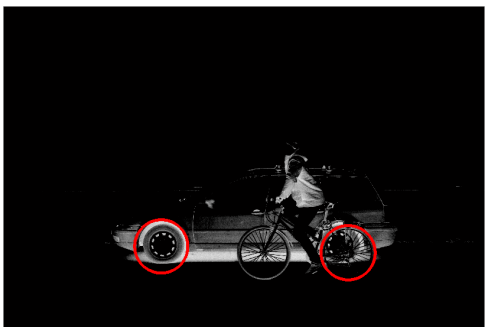
Hough-transformen finner hjulene på sykkelen som de to mest fremtredende sirklene på bildene Sykkel6-Sykkel10. I gjennomsnitt finner detektoren 25017.6 egenskapspunkter og bruker 140 sekunder for hvert bilde. Se tabell 5.9 og resultatene i 5.5.

GeoT = 0.60·n_{maks} og T = 57:

Hough-transformen finner hjulene på sykkelen som de to mest fremtredende sirklene fra bildene Sykkel6-Sykkel10. Punktene er redusert med 1817 punkter og tiden med -1.364 i gjennomsnitt. Se tabell 5.10 og resultatene i 5.6.

GeoT = 0.59·n_{maks} og T = 57:

Hough-transformen finner ikke hjulene på alle bildene, dermed vil en gå tilbake til geometrisk terskel 0.6 ·n_{maks}. Bildene hvor Hough-transformen finner sirkler utenfor sykkelen er vist i figur 4.21. Resten av resultatene ved disse tersklene er vist i 5.7.



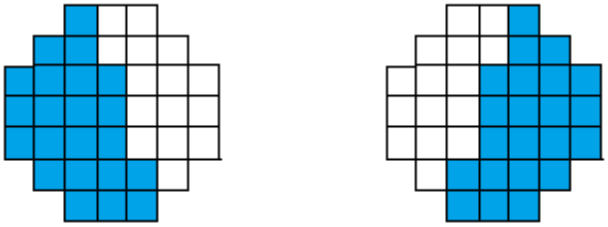
(a) Sykkel6

(b) Sykkel8

(c) Sykkel9

Figur 4.21: Figuren viser bildene hvor Hough-transformen ikke finner sykkelhjulene som de to mest fremtredende sirklene i bildet. Dette er ved geometrisk terskel lik 0.59·n_{maks}

Den optimale geometriske terskelen er satt til 0.60·n_{maks} ut i fra denne testingen. Dette vil tilsvare 22 punkt innen den sirkulære masken som må være tilnærmet like for at et egenskapspunkt skal bli funnet. Dette passer den buede kanten på sykkelhjulet. En approksimasjon av masken er vist i figur 4.22.



(a) (b)

Figur 4.22: Figuren viser hvordan masken kan se ut når geometrisk terskel er $0.6 \cdot n_{maks}$.

Endring av maskestørrelse

Det er testet hvordan endring av maskestørrelsen påvirker deteksjonen av egenskapspunkter. Dette er vist i figur 5.6.

Maskeradius	Antall egenskapspunkter	Tid[s]	Tid per egenskapspunkt[s]
5	53077	281.13	0.0053
7	81709	414.11	0.0051
9	108023	594.91	0.0055
11	136073	817.85	0.0060

Tabell 4.7: Antall egenskapspunkter og tidsbruk ved endring av maskestørrelse

Antall egenskapspunkter øker ifølge 5.6. Før endringen av masken finner detektoren mange punkter og det er ønskelig å redusere disse. Den opprinnelige maskeradiusen er 3 piksler. Endring av maskestørrelsen vil ikke gi fordeler til SUSAN-detektoren. Økt antall punkter vil gi Hough-transformen flere punkter å lete over, og det vil føre til at deteksjonen blir tregere.

4.5 Tilpasning til sykkeltelling

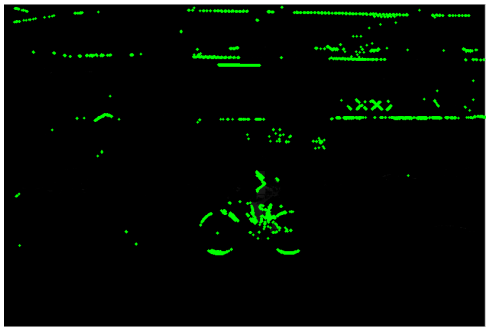
For å kunne verifisere at det er en sykkel som er funnet er det lagt til noen kriterier til sirklene funnet ved Hough-transformen. Gitt radiusen på hjulet og avstanden mellom hjulenes senter i x-retning kan et størrelsesforhold settes opp. Dette størrelsesforholdet vil kunne brukes til verifisering av sykkelhjulene. Utgangspunktet til dette størrelsesforholdet er laget ut i fra 10 trafikkbilder 5.1 og hentet ut ved hjelp av matlab sin funksjon *imtool*. Bildesettet 5.1 er tatt med eget kamera og stativ, avstanden mellom kamera og scene vil dermed være konstant.

Tabellen 5.6 viser at forholdet mellom avstand i x-retning og radius ligger mellom 3.59 og 3.02. Dermed settes det opp at avstanden mellom hjulene må være $3.3 \cdot radius$ i tillegg til en variasjon på 0.3 på størrelsesforholdet.

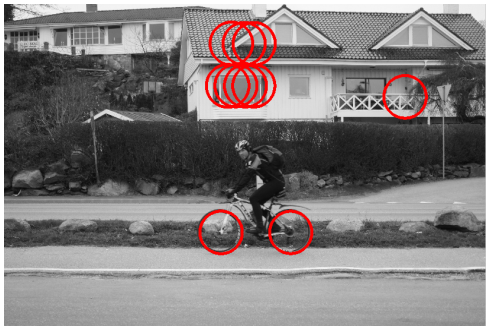
Vinkelen mellom sirklenes senter bør være liten. Sykkelhjulenes posisjon vil være horisontale i forhold til hverandre. Det kan være noe forskjell grunnet Hough-transformens deteksjon av sirklene. Ulikheten i piksler er satt til å kunne være 30. Ved test av avstand i y-retning var 19 lengste avstand 5.7. Grunnen til at den er satt høyere enn dette er stor variasjon blant avstandene.

Ved å legge til disse to kriteriene vil sykkelen bli funnet tross av at sykkelhjulene ikke er de to mest fremtredende sirklene gitt egenkapspunktene fra SUSAN-detektoren. Figur 4.23 viser et resultat av dette.

Delfigur 4.23b viser alle de 10 mest fremtredende sirklene funnet av Hough-transformens sirkeldeteksjon. Delfigur 4.23c viser hvilke to av de 10 sirklene som er mest fremtredende. Tilslutt vises 4.23d hvor sykkelen er funnet gitt størrelsesforhold og vinkel mellom sirklene.

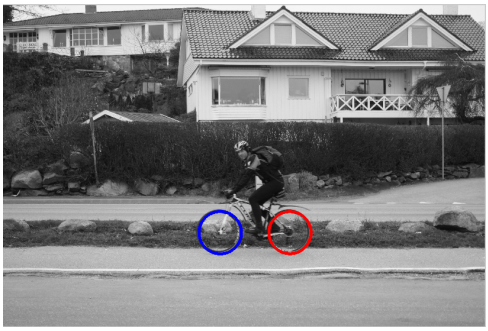


(a) Egenskapspunktene funnet ved SUSAN-detektoren



(b) De 10 mest fremtredende sirklene funnet ved Hough-transformen.

(c) De 2 mest fremtredende sirklene funnet ved Hough-transformen.



(d) Sykkel detektert gitt størrelsesforhold og avstand mellom sirklene.

Figur 4.23: Figuren viser hvordan verifikasjonen av sykkelen fungerer. Dette er gitt fremtredende sirkler funnet av Hough-transformen og egenskapspunkter av SUSAN-detektoren

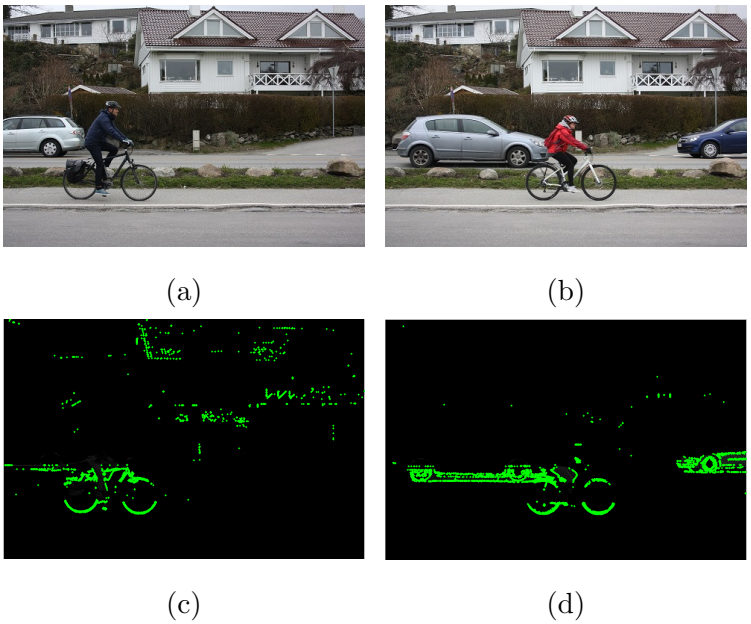
4.6 Test av detektor på trafikkbilder

For å teste detektoren mot realistiske bilder har det blitt tatt en del bilder av en sykkelvei hvor det er en trafikkert vei i bakgrunnen.

Grunnet at mengden støy har økt vil bakgrunnssubtraksjonen (selv med vekting av piksler) gi større endringer i pikslene ved bilene enn ved sykkelen. Ved normalisering av disse verdiene (som gjort ved de enkle bildene) vil Hough-transformen finne de mest fremtredende sirklene ved bilene. Grunnet dette er det valgt å ikke bruke normalisering ved disse verdiene. Dermed må gråterskelverdien settes ned i forhold til test på tidligere bilder. Bildene blir testet med den optimaliserte geometriske terskelen, $0.6 \cdot n_{maks}$.

SUSAN-detektoren bruker i snitt 134.8 sekunder per bilde (se tabell **??** i vedlegg). I tillegg finner detektoren i gjennomsnitt 4691.3 egenskapspunkter i hvert bilde med de gitte terskelverdiene. Mange av punktene som blir funnet i denne sammenheng sees på som støy og er ikke en del av sykkelen. Grunnen til at punkter blir funnet utenfor objektet er endringer i bildet mellom bilderammene. Dette skapes av vær og vind, det skal lite til å endre intensitetsverdiene i bildet. Fra resultatene av egenskapsdeteksjon ved 10 testbilder 5.1 er det stor ulikhet i hvor støypunktene blir funnet og antall punkter funnet 5.12. Resultatet av de 10 testbildene finner man her: 5.9 og 5.8.

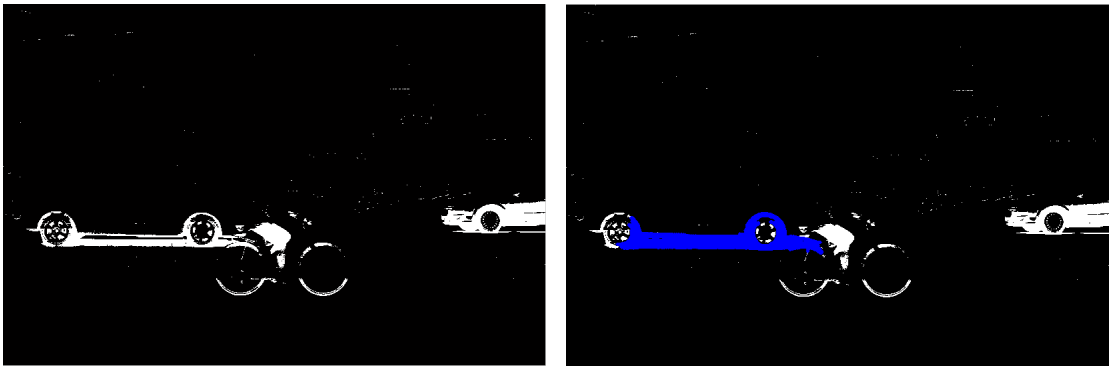
I figur 4.24 vises to bilder av SUSAN-deteksjonen på trafikkbildene. De grønne punktene representerer punktene funnet ved deteksjonen. Forskjellen i antall egenskapspunkter funnet på taket i bakgrunnen er stor. Bakgrunnssubtraksjonen er den samme samt terskelverdier. Dette viser hvor mye som kan endres fra bilderamme neste. Alle trafikkbildene er tatt med kun minutters mellomrom, dermed vil små endringer gjøre stor forskjell i subtraksjonen.



Figur 4.24: Endring i støypunkter mellom to bilderammer. Grønne punkter representerer egenskapspunter funnet.

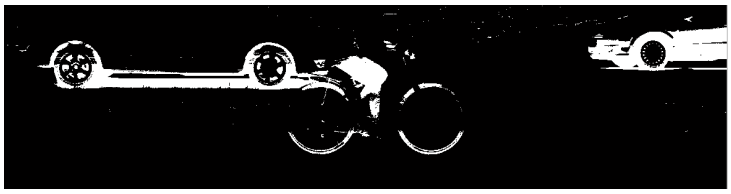
4.6.1 Utklipp av interessant område

Syklisten vil kun oppta en liten del av bildet. For å redusere støypunktene funnet ved deteksjonen samt tiden vil det være en fordel å klippe ut de delene av bildet en er interessert i. Dette kan gjøres ved å først gjøre bildet om til binært deretter lete over bildet etter området med flest hvite piksler sammenhengende (dette er utført med matlab funksjonen regionprops, med parameteret BoundingBox). Etter bakgrunnssubtraksjonen vil dette tilsvare sykkelen eller andre objekter på veien. Derfor klippes hele x-aksen tross største objekt deretter klippes y-retning etter det største objektet i bildet. Hvordan utklippingen utføres er vist i figur 4.25.



(a) Det binære bildet etter bakgrunnssubtraksjon.

(b) Blå punkter representerer største objektet funnet i bildet.

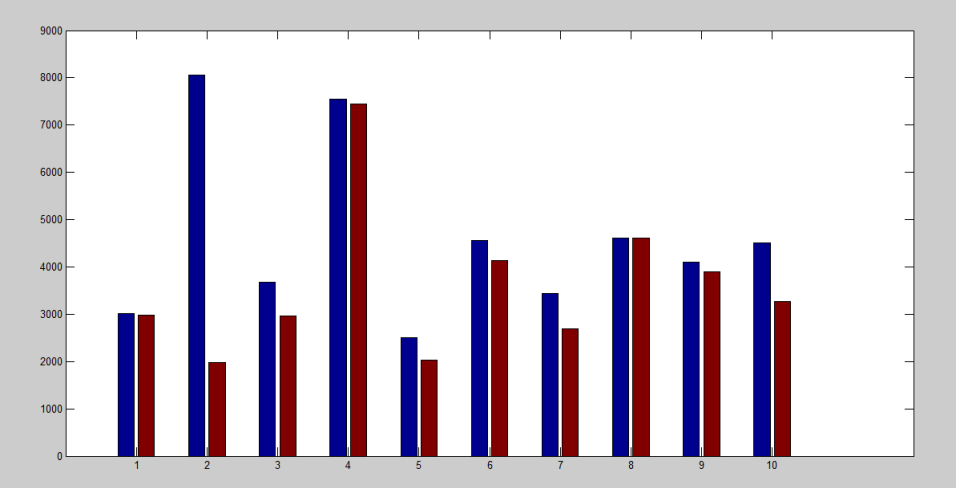


(c) Det utklippede bildet.

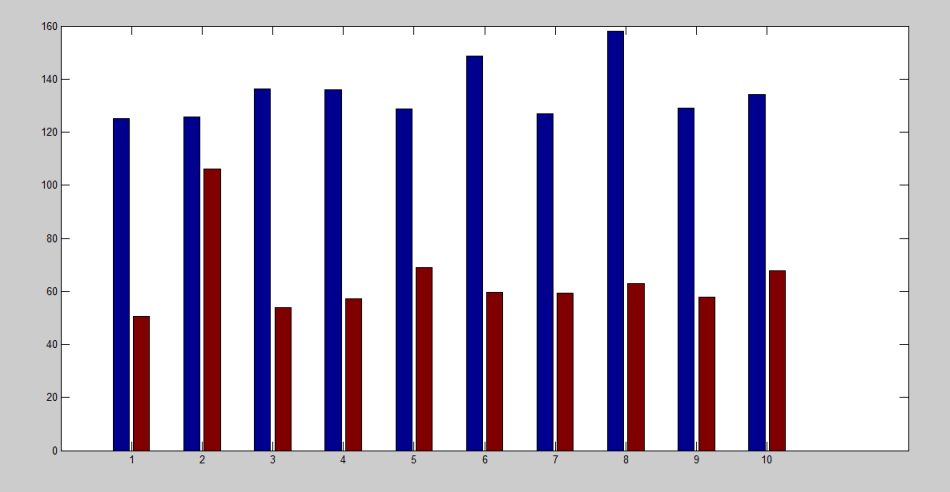
Figur 4.25: Utførelsen av utklipping av interessant område

Resultatet av utklipp på de 10 trafikkbildene er vist i 5.12 og 5.13. Gjennomsnittstiden detektoren bruker på bildene etter utklippet er 64.43 sekunder. Dette er en 51.78 % redusering av tiden. I tillegg vil den kun detektere 3597 punkter i gjennomsnitt per bilde, som gir en redusering med prosentandel 18.04%. Dette er en stor forbedring. Hjulene fra de 10 testbildene detekteres ved hjelp av Hough-transformen på alle bildene tross

utklippingen. En figur som viser reduseringen er vist i figur 4.26 og 4.27. Resultatbildene kan man se i referanse til bildemappe?!.



Figur 4.26: Grafen viser reduseringen i antall punkter funnet med og uten utklippingen av det interessante området. De røde søylene viser uten utklipp, mens de blå viser med. X-aksen vil kun være nummeret på bildet, mens y-aksen representerer antall punkter funnet ved deteksjon.

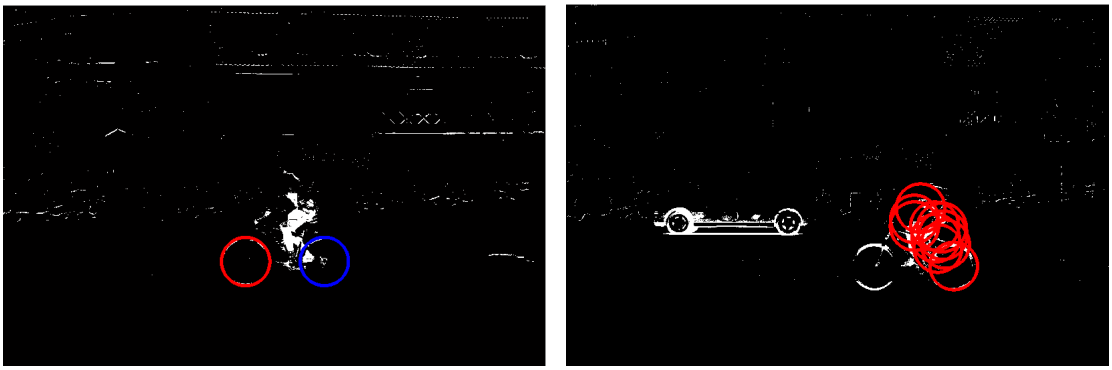


Figur 4.27: Grafen viser redusering i tid funnet med og uten utklippingen av det interessante området. De røde søylene viser uten utklipp, mens de blå viser med. X-aksen viser bildenummeret, mens y-aksen representerer tiden brukt ved deteksjon (gitt i sekunder).

4.6.2 Hough-transformen testet på binære bilder

Et alternativ til å bruke egenskapsdeteksjon for å finne sykkelen er å bruke Hough-transformen rett på de resulterende bildene fra bakgrunnssubtraksjonen (binære bilder). De 10 binære trafikkbildene er testet mot Hough-transformen og verifiseringen av sykkelen.

Resultatet viser at ved 3 av 10 testbilder finner ikke Hough-transformen hjulene (Se 5.12 og 5.13). De hvite pikslene utgjør personen på sykkelen gir store utslag ved sirkeldeteksjonen. Selv ved deteksjon av de 10 mest fremtredende sirklene i bildet finner ikke Hough-transformen begge hjulene. Dette er vist i figur 4.28.

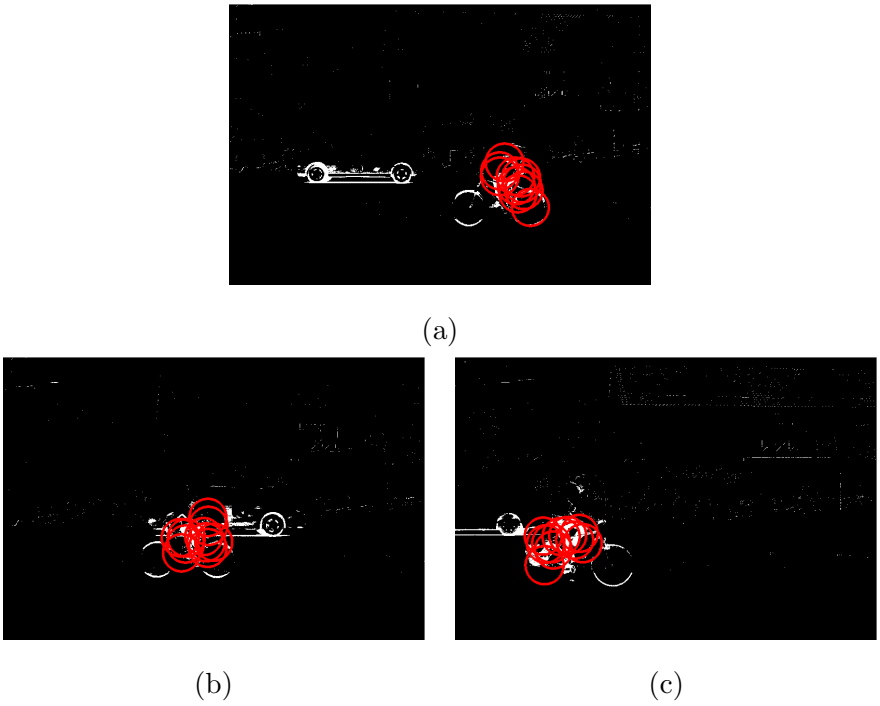


(a) Bildet 'TrafikkSykkel2' med Hough-transformen og verifisering av sykkelen.

(b) Bildet 'TrafikkSykkel3' med Hough-transformen. Finner ikke hjulene ved verifiseringen av sykkelen.

Figur 4.28: Fremstilling av Hough-transformen utført direkte på binære bilder.

Egenskapspunktene blir funnet på syklisten i stedet for sykkelen og dette vil ikke endres ved utklipp (se figur 4.29).



(a)

(b) (c)

Figur 4.29: Hough-transformen testet på binære bilder.

Med en feilprosent på 33.33 % vil det ikke være en fordel å bruke Hough-transformen på binære bilder og utelukke egenskapsdeteksjonen. Den vil redusere tiden brukt per bilde, men dette vil ikke veie opp for robustheten som reduseres.

Kapittel 5

Konklusjon

I denne oppgaven har det blitt sett på muligheten for å bruke egenskapsdetektorer til objektgjenkjenning av sykler. Det har blitt testet tre ulike detektorer, BRISK-, SUSAN- og SURF-detektoren, opp mot objektgjenkjenningen og konkludert med at SUSAN-detektorens egenskapspunkter passer beskrivelsen av sykkelhjulet best. Bruksområdet kan brukes til telling av syklister ved sykkelstier. Hovedkonklusjonen er at SUSAN-detektoren kan brukes til dette.

Optimaliseringen av SUSAN-detektoren reduserer tidsforbruket og fjerner punkter som ikke er relevante. I tillegg fungerer sykkel-verifiseringen og utklipp av interessant område slik som forventet. Detektoren med disse tilleggene gir et godt resultat.

Selv med reduseringen av tidsforbruket er SUSAN-detektoren treg i forhold til BRISK-detektoren som er testet tidligere i oppgaven. Men sett ut i fra tidsrammen man hadde trengt ved telling av syklister er dette akseptabelt.

5.1 Videre arbeid

Sykkelhjulenes egenskapspunkt ligner mer kanter enn hjørner. Dette gir SUSAN-detektoren fordeler i forhold til BRISK og SURF i og med at det kan reguleres. Derfor ville det

vært hensiktsmessig å velge ut konkurrerende detektorer som fokuserer på kanter og ikke hjørner slik som BRISK og SURF. BRISK og SURF opptrer robust i noen av testene, men grunnet egenskapspunkter funnet i irrelevante områder blir sirkeldeteksjonen vanskelig.

Objektgjenkjenning av andre objekter (biler, gående, andre kjøretøy osv.) knyttet til trafikken ville vært interessant. Kriterier knyttet til disse objektene kunne vært lagt til og gjenkjenningen hadde gitt et større 'trafikkbilde'.

Bibliografi

[1] Z. K. Li Jia, Sheng Ye-hua and D. Ping. (2014, May) Image matching of variable circular domain compass features.

[2] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detetction based on the accelerated segment test," 2010.

[3] Wikipedia. Binary large object [05.02.2015]. [Online]. Available: http://en.wikipedia.org/wiki/Binary_large_object

[4] ——. Ridge detection [09.02.2015]. [Online]. Available: http://en.wikipedia.org/wiki/Ridge_detection

[5] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding 110 (2008)*, 2007.

[6] Wikipedia. Slam [23.01.2015]. [Online]. Available: http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[7] C. G. Harris and M.Stephens, "A combined corner and edge detector," *In 4th Alvey Vision Conference*, 1988.

[8] T. Lindeberg, "Scale-space for discrete signals," *IEE Transactions on pattern analysis and machine intelligence, vol. 12*, March 1990.

[9] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," *IEEE International Conference on Computer Vision*, 2011.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *In Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

[12] S. T. Pathan, S. S. Waghmare, P. P. Khadse, P. P. Shukla, and S. A. Wawre, "A survey paper on a novel approach for image classification based on susan low level image processing algorithm from real time video," *International Journal of Scienctific and Technology Research Volume 3, Issue 2*, February 2014.

[13] V. Khongpit, U. phangphol, K. Kasemsan, and C. Srisa-an, "2d gun's type classification using edge detection algorithm and susan low level image processing algorithm from real time video," *The 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012)*, 2012.

[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," Master's thesis, Cambridge University, 2006.

[15] Wikipedia. 'midpoint circle algorithm' [11.02.2015]. [Online]. Available: http://en.wikipedia.org/wiki/Midpoint_circle_algorithm

[16] A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, J. Ascenso, and R. Cilla, "Evaluation of low-complexity visual feature detectors and descriptors," *Programme for Research of the European Commission, under FET-Open grant, number:296676.*, 2013.

[17] S. Smith and J. Brady, "Susan - a new approach to low level image processing," *Int. Journal of Computer Vision*, May 1997.

[18] W. Muyun and H. Mingyi, "Image feature detection and matching based on susan method," *Proceedings of the First International Conference on Innovative Computing, Information and Control*, 2006.

[19] S. Smith, "Feature based image sequence understanding," *D.Phil. thesis, Robotics Research Group, Department of Engineering Science, Oxford University*, 1992.

[20] Wikipedia. Hessian matrix [30.03.2015]. [Online]. Available: http://en.wikipedia.org/wiki/Hessian_matrix

[21] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Accepted Conference on Computer Vision and Pattern Recognition*, 2001.

[22] M. Brown and D. Lowe, "Invariant features from interest point groups," *BMVC*, 2002.

[23] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach (Second edition)*, S. Mukherjee and A. K. Bhattacharjee, Eds. Pearson, 2012.

[24] D. Young. Hough-transform [06.06.2015]. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/26978-hough-transform-for-circles

Appendix

A - Tabeller

Tabell 5.1: SURF-detektoren ved lysendring (0-25%). Bilde: Sykkel1

Lysendring [%]	Antall egenskapspunkter funnet	Deteksjonstid [s]	Prosent av punkter
0.0 %	752	4.83	100 %
2.5 %	750	4.15	99.73%
5.0 %	705	6.27	93.75 %
7.5%	715	4.27	95 %
10.0 %	661	4.45	87.9%
12.5 %	657	4.37	87.37 %
15.0 %	630	6.28	83.78 %
17.5 %	624	4.32	82.98 %
20.0 %	598	4.41	79.52 %
22.5 %	597	4.28	79.39 %
25.0 %	558	4.30	74.2 %

Tabell 5.2: BRISK-detektoren ved lysendring (0-25%). Bilde: Sykkel1

Lysendring [%]	Antall egenskapspunkter funnet	Deteksjonstid [s]	Prosent av punkter
0.0 %	533	0.440	100 %
2.5 %	553	0.361	103%
5.0 %	481	0.330	90.24 %
7.5%	487	0.423	91.37 %
10.0 %	434	0.412	81.43%
12.5 %	444	0.397	83.30 %
15.0 %	406	0.394	76.17 %
17.5 %	405	0.414	75.99 %
20.0 %	385	0.415	72.23 %
22.5 %	345	0.404	64.72 %
25.0 %	322	0.330	60.41 %

Tabell 5.3: SUSAN-detektoren ved lysendring (0-25%). Bilde: Sykkel1

Lysendring [%]	Antall egenskapspunkter funnet	Deteksjonstid [s]	Prosent av punkter
0.0 %	9828	193.98	100 %
2.5 %	8875	193.88	90.30%
5.0 %	8321	194.55	84.67 %
7.5%	8578	0.188.94	87.70 %
10.0 %	7408	207.35	75.38%
12.5 %	7616	208.18	77.49 %
15.0 %	6657	183.92	67.74 %
17.5 %	6717	209.97	68.35 %
20.0 %	6452	202.75	65.65 %
22.5 %	5982	191.39	60.87 %
25.0 %	5626	181.65	57 %

Tabell 5.4: Endring av gråverditerskel

Gråverdi	Antall egenskapspunkter funnet	Deteksjonstid [s]	Prosent av punkter
58	25910		%
59	24598		%
60	23375		%
61	22179		%
62	21066		%
63	20043		%
64	19062		%
65	18065		%
66	17073		%
67	16190		%
68	15404		%

Tabell 5.5: Endring av geometrisk terskel

Geometrisk terskel	Antall egenskapspunkter funnet	Deteksjonstid [s]	Prosent av punkter
61	23603		%
60	21811		%
59	20328		%
58	18954		%
57	17814		%
56	16616		%
55	15658		%
54	14759		%

Tabell 5.6: Tabell som viser størrelsesforholdet mellom radius og avstand mellom hjulene.

Bilde	Avstand [piksler]	Radius [piksler]	Størrelsesforhold
TrafikkSykkel1	558	174	3.21
TrafikkSykkel2	540	162	3.33
TrafikkSykkel3	528	175	3.02
TrafikkSykkel4	600	168	3.57
TrafikkSykkel5	570	174	3.28
TrafikkSykkel6	540	162	3.33
TrafikkVanskelig1	534	163	3.28
TrafikkVanskelig2	564	170	3.32
TrafikkVanskelig3	552	165	3.35
TrafikkVanskelig4	600	167	3.59

Tabell 5.7: Tabell som viser avstand i y-retning mellom sykkelhjulenes senter.

Bilde	Avstand i y-retning [piksler]	vinkel [grader]
TrafikkSykkel1	13	1.33
TrafikkSykkel2	1	0.11
TrafikkSykkel3	1	0.11
TrafikkSykkel4	4	0.38
TrafikkSykkel5	19	1.9
TrafikkSykkel6	0	0
TrafikkVanskelig1	5	0.54
TrafikkVanskelig2	10	1.02
TrafikkVanskelig3	6	0.62
TrafikkVanskelig4	14	1.34

Tabell 5.8: SUSAN-detektoren ved utklipp av objektene, T=30 og $geoT = 0.6 \cdot n_{maks}$.

Bilde	Antall egenskapspunkter funnet	deteksjonstid [s]
TrafikkSykkel1	2972	50.43
TrafikkSykkel2	1972	106.07
TrafikkSykkel3	2955	53.99
TrafikkSykkel4	7441	57.19
TrafikkSykkel5	2029	69.05
TrafikkSykkel6	4130	59.53
TrafikkVanskelig1	2695	59.26
TrafikkVanskelig2	4608	63.06
TrafikkVanskelig3	3902	57.8
TrafikkVanskelig4	3266	67.87

Tabell 5.9: Kombinasjon av geometrisk terskel og gråverditerskel, T=57 og $geoT = 0.61 \cdot n_{maks}$.

Bilde	Antall egenskapspunkter funnet	deteksjonstid [s]
Sykkel6	31139	143.92
Sykkel7	23603	141.36
Sykkel8	24156	138.31
Sykkel9	20842	139.54
Sykkel10	25348	137.15

Tabell 5.10: Kombinasjon av geometrisk terskel og gråverditerskel, T=57 og $geoT = 0.60 \cdot n_{maks}$.

Bilde	Antall egenskapspunkter funnet	deteksjonstid [s]
Sykkel6	28894	140.25
Sykkel7	21811	142.60
Sykkel8	22414	139.48
Sykkel9	19359	142.23
Sykkel10	23525	142.26

Tabell 5.11: Kombinasjon av geometrisk terskel og gråverditerskel, T=57 og $geoT = 0.59 \cdot n_{maks}$.

Bilde	Antall egenskapspunkter funnet	deteksjonstid [s]
Sykkel6	26753	138.5
Sykkel7	20328	145.05
Sykkel8	20742	139.48
Sykkel9	17980	141.66
Sykkel10	21805	139.89

Tabell 5.12: SUSAN-detektoren uten utklipp av objekter, T=30 og $geoT = 0.6 \cdot n_{maks}$.

Bilde	Antall egenskapspunkter funnet	deteksjonstid [s]
TrafikkSykkel1	3018	125.26
TrafikkSykkel2	8051	125.78
TrafikkSykkel3	3684	136.38
TrafikkSykkel4	7548	135.15
TrafikkSykkel5	2505	128.71
TrafikkSykkel6	4559	148.74
TrafikkVanskelig1	3435	127.05
TrafikkVanskelig2	4609	157.94
TrafikkVanskelig3	4101	128.97
TrafikkVanskelig4	4503	134.06

Tabell 5.13: Prosent spart ved utklipp av objekter før deteksjon av egenskapspunkter, T=30 og geoT = $0.6 \cdot n_{maks}$.

Bilde	Prosent spart i fohold til antall punkter	Prosent i forhold til tid
TrafikkSykkel1	1.5	59.74
TrafikkSykkel2	75.5	15.67
TrafikkSykkel3	19.79	60.41
TrafikkSykkel4	1.42	57.68
TrafikkSykkel5	19	46.35
TrafikkSykkel6	9.4	59.98
TrafikkVanskelig1	21.54	53.36
TrafikkVanskelig2	0.022	60.07
TrafikkVanskelig3	4.85	55.18
TrafikkVanskelig4	27.47	49.37

B - Bildesett

B1, Enkle bilder

Bildesettet er tatt med speilreflekskamera, Canon 1000D. Dette er et kamera med bildeoppløsning 10.1 Mp.

(a) sykkel1

(b) sykkel2

(c) sykkel3

(d) sykkel4

(e) sykkel5

(f) sykkel6

(g) sykkel7

(h) sykkel8

(i) sykkel9

(j) sykkel10

Figur 5.1: Enkle bilder med sykkel

B2 - Trafikkbilder

Bildesettet er tatt med speilrefleksskamera, Canon 1000D. Dette er et kamera med bildeoppløsning 10.1 Mp. Bildene er tatt ved Hafrsfjord hvor det allerede er installert en induktiv sykkelteller.

(a) TrafikkSykkel1

(b) TrafikkSykkel2

(c) Trafikkykkel3

(d) TrafikkSykkel4

(e) TrafikkSykkel5

(f) TrafikkSykkel6

(g) TrafikkVanskelig1

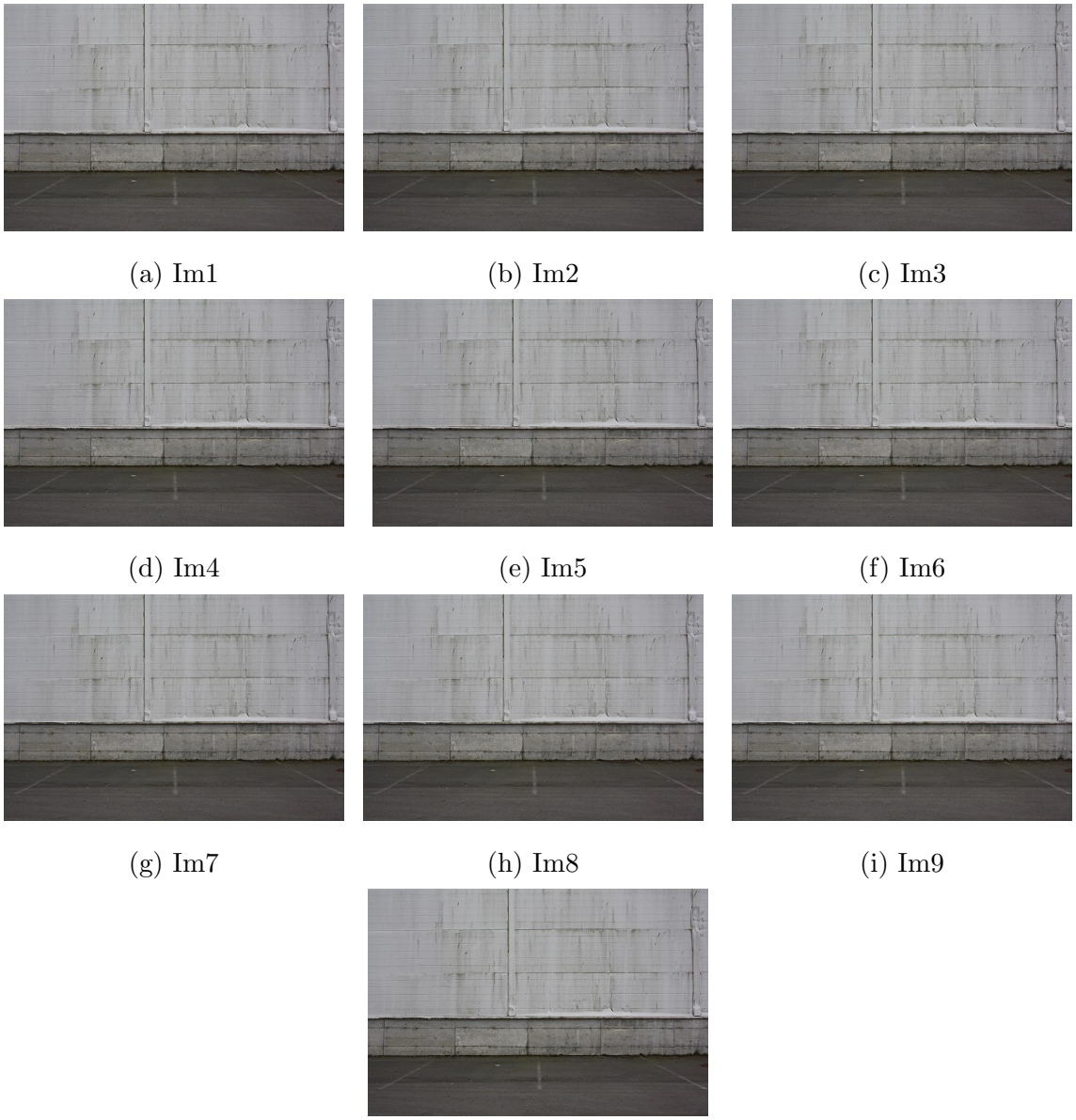
(h) TrafikkVanskelig2

(i) TrafikkVanskelig3

(j) TrafikkVanskelig4

Figur 5.2: Sykkelbilder i trafikken

B3 - Enkle bakgrunnsbilder



(a) Im1 (b) Im2 (c) Im3

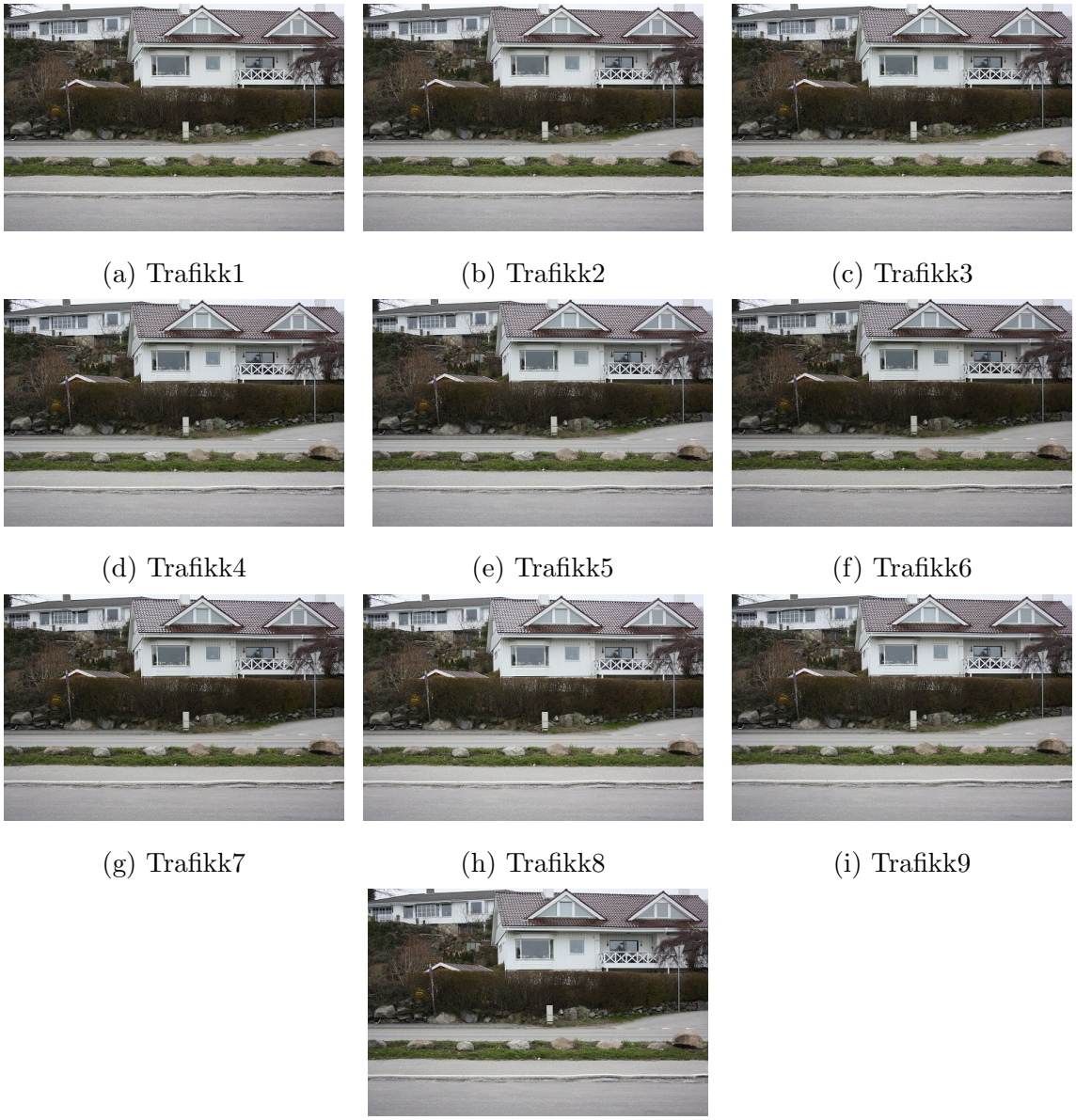
(d) Im4 (e) Im5 (f) Im6

(g) Im7 (h) Im8 (i) Im9

(j) Im10

Figur 5.3: Enkle bakgrunnsbilder

B4 - Trafikk-bakgrunnsbilder



(a) Trafikk1 (b) Trafikk2 (c) Trafikk3

(d) Trafikk4 (e) Trafikk5 (f) Trafikk6

(g) Trafikk7 (h) Trafikk8 (i) Trafikk9

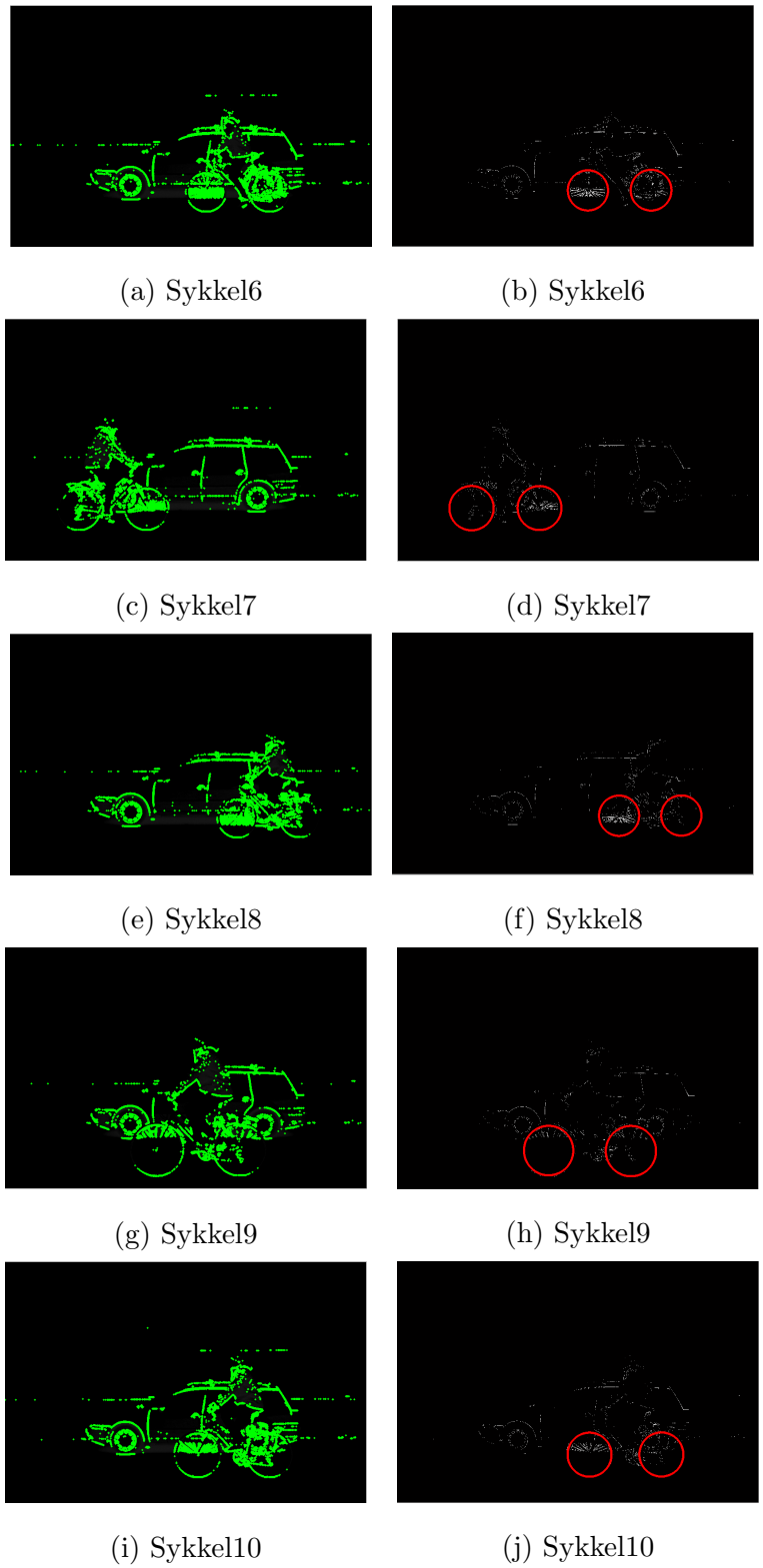
(j) Trafikk10

Figur 5.4: Trafikk-bakgrunnsbilder

C - Resultatbilder

C1 - Kombinasjon av gråverditerskel og geometrisk terskel

geo=61, T=57



(a) Sykkel6 (b) Sykkel6

(c) Sykkel7 (d) Sykkel7

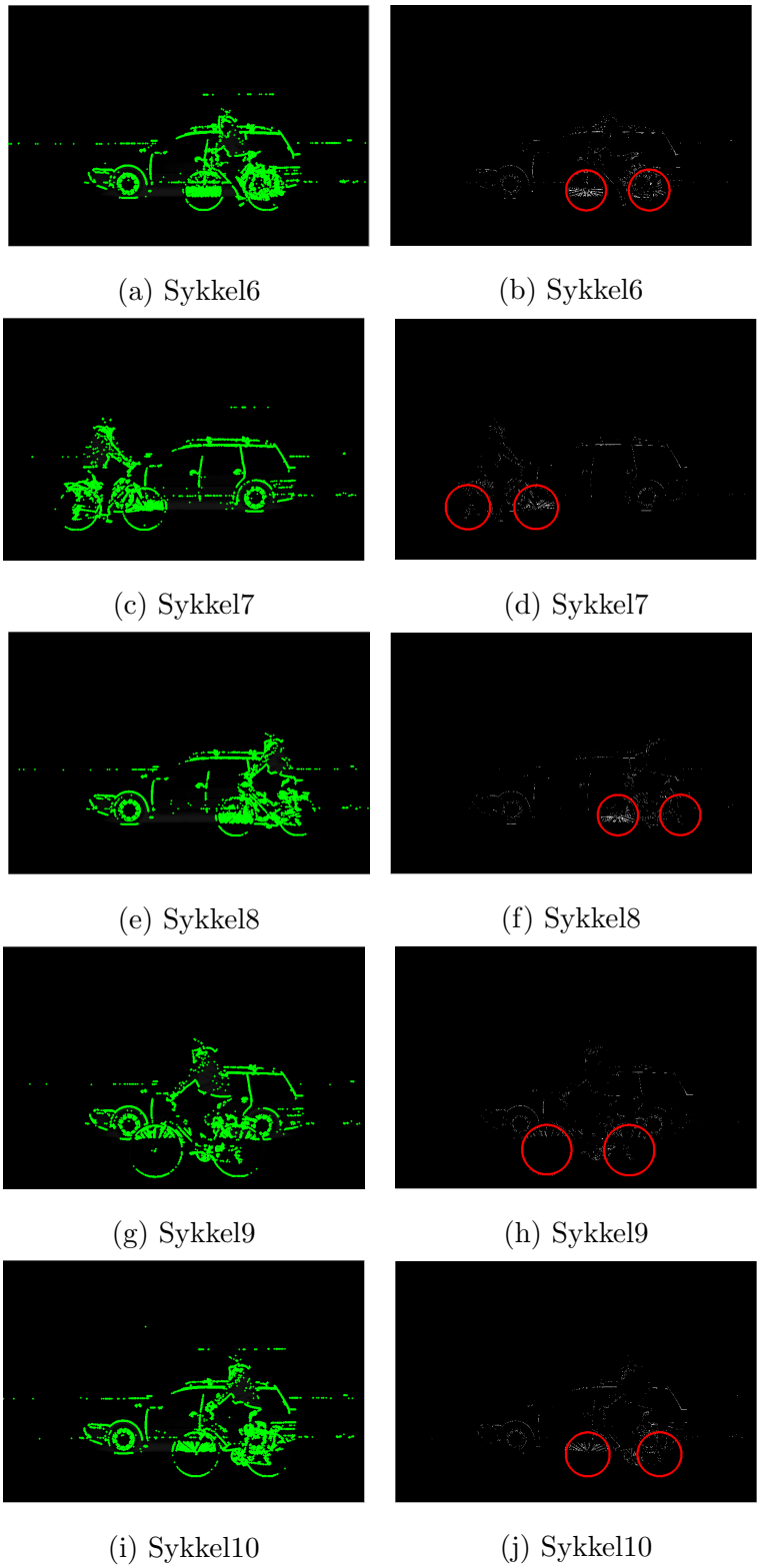
(e) Sykkel8 (f) Sykkel8

(g) Sykkel9 (h) Sykkel9

(i) Sykkel10 (j) Sykkel10

Figur 5.5: Geometrisk terskel lik 61 og gråverditerskel lik 57.

geo=60, T=57



(a) Sykkel6 (b) Sykkel6

(c) Sykkel7 (d) Sykkel7

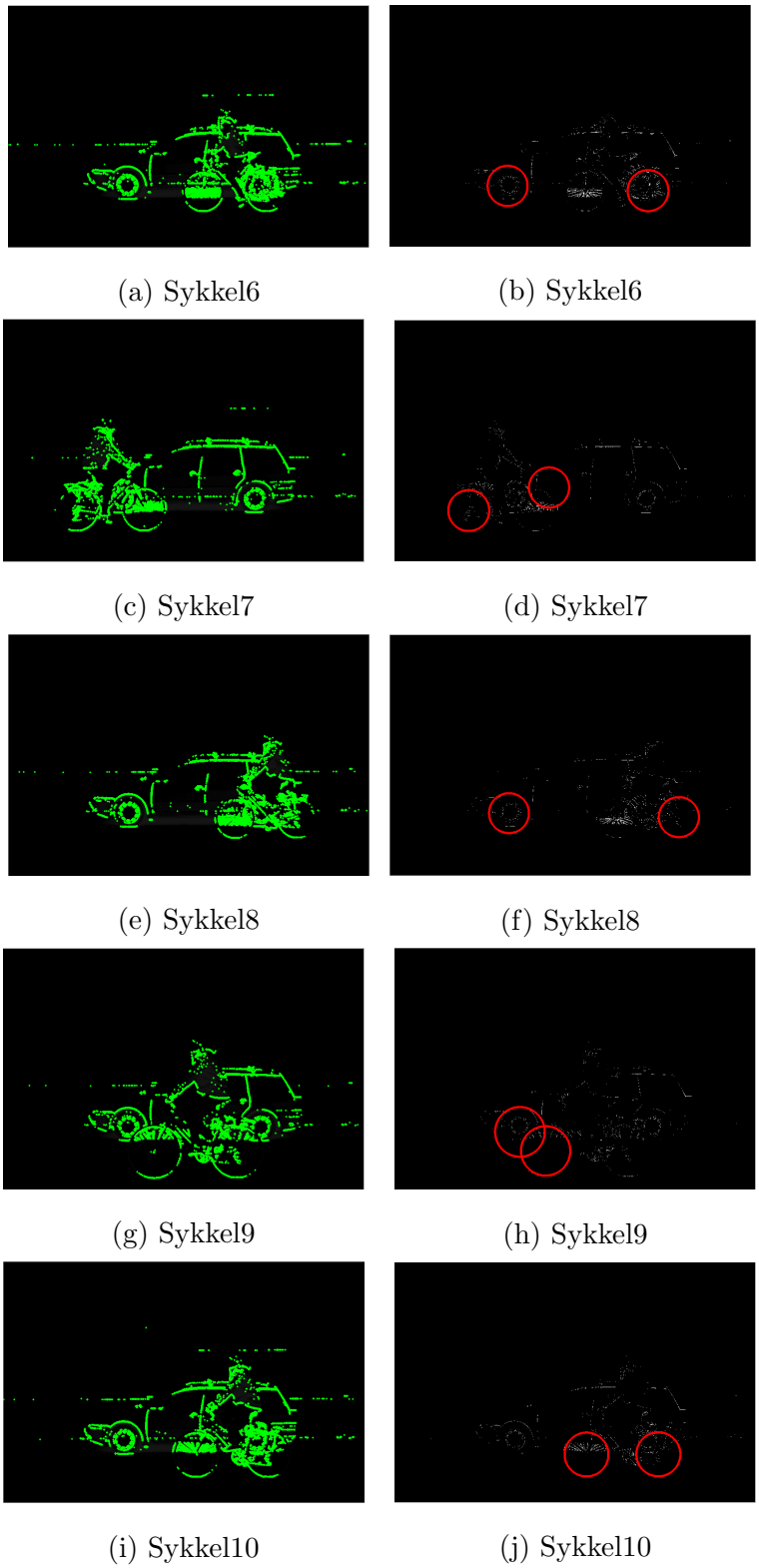
(e) Sykkel8 (f) Sykkel8

(g) Sykkel9 (h) Sykkel9

(i) Sykkel10 (j) Sykkel10

Figur 5.6: Geometrisk terskel lik 60 og gråverditerskel lik 57.

geo=59, T=57



(a) Sykkel6 (b) Sykkel6

(c) Sykkel7 (d) Sykkel7

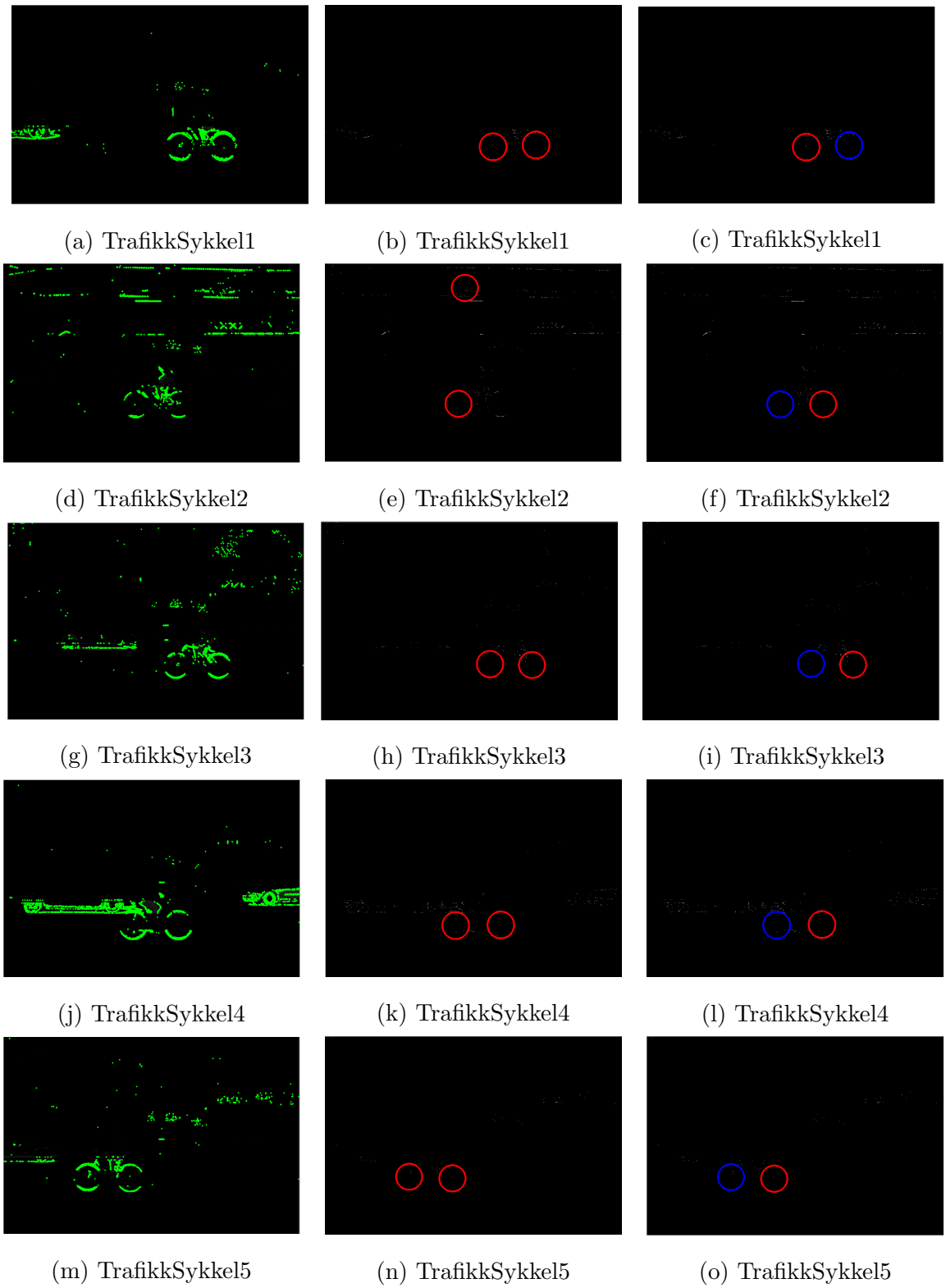
(e) Sykkel8 (f) Sykkel8

(g) Sykkel9 (h) Sykkel9

(i) Sykkel10 (j) Sykkel10

Figur 5.7: Geometrisk terskel 59 og gråverditerskel 57.

C2 - SUSAN-detektoren testet på trafikkbilder



(a) TrafikkSykkel1 (b) TrafikkSykkel1 (c) TrafikkSykkel1

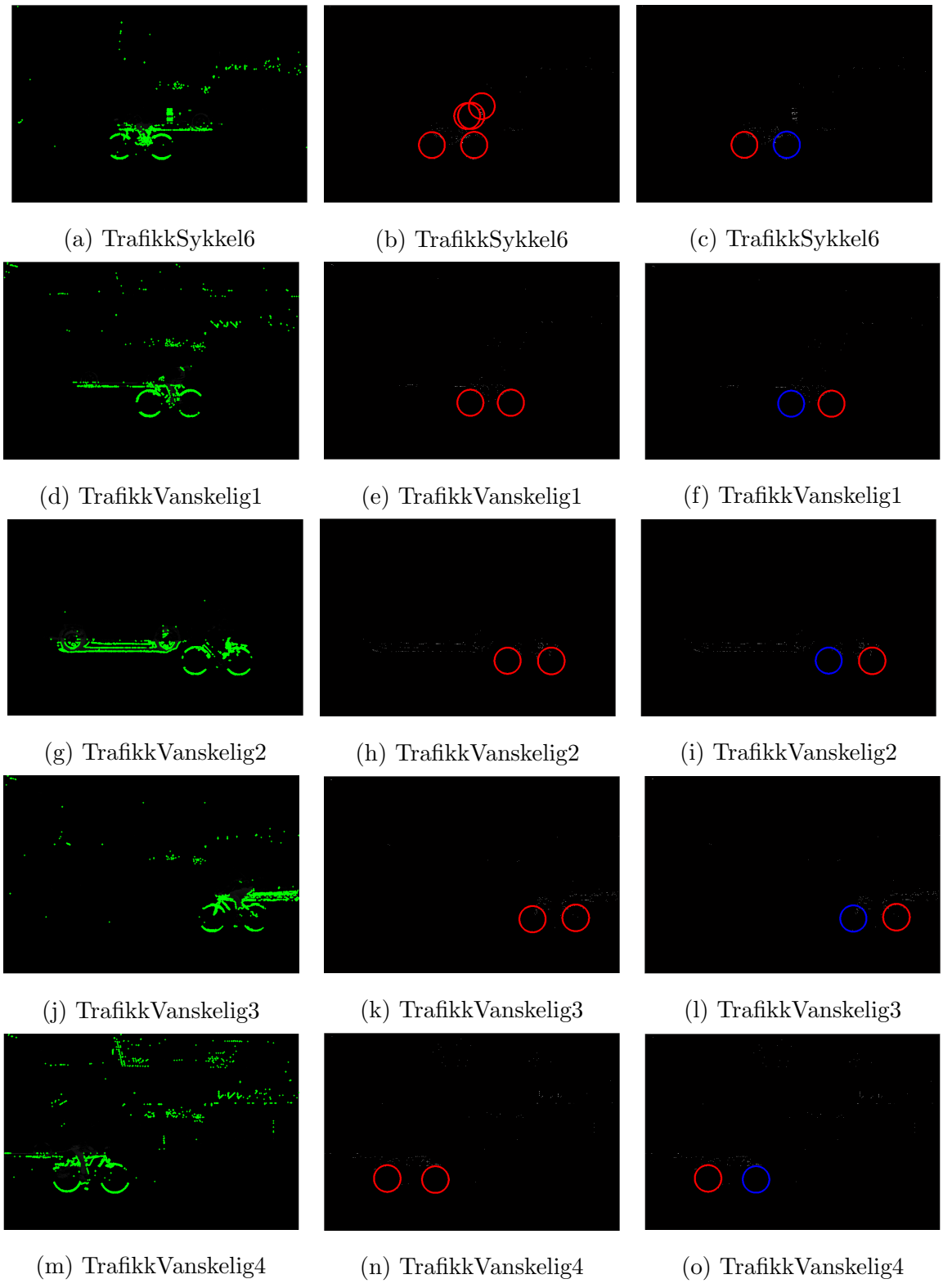
(d) TrafikkSykkel2 (e) TrafikkSykkel2 (f) TrafikkSykkel2

(g) TrafikkSykkel3 (h) TrafikkSykkel3 (i) TrafikkSykkel3

(j) TrafikkSykkel4 (k) TrafikkSykkel4 (l) TrafikkSykkel4

(m) TrafikkSykkel5 (n) TrafikkSykkel5 (o) TrafikkSykkel5

Figur 5.8: Geometrisk terskel 60 og gråverditerskel lik 30.



(a) TrafikkSykkel6 (b) TrafikkSykkel6 (c) TrafikkSykkel6

(d) TrafikkVanskelig1 (e) TrafikkVanskelig1 (f) TrafikkVanskelig1

(g) TrafikkVanskelig2 (h) TrafikkVanskelig2 (i) TrafikkVanskelig2

(j) TrafikkVanskelig3 (k) TrafikkVanskelig3 (l) TrafikkVanskelig3

(m) TrafikkVanskelig4 (n) TrafikkVanskelig4 (o) TrafikkVanskelig4

Figur 5.9: Geometrisk terskel 60 og gråverditerskel lik 30.

C3 - Utklippet område

Punkter funnet, to mest fremtredende sirkler og verifiserte sirkler.

(a) TrafikkSykkel1 (b) TrafikkSykkel1 (c) TrafikkSykkel1

(d) TrafikkSykkel2 (e) TrafikkSykkel2 (f) TrafikkSykkel2

(g) TrafikkSykkel3 (h) TrafikkSykkel3 (i) TrafikkSykkel3

(j) TrafikkSykkel4 (k) TrafikkSykkel4 (l) TrafikkSykkel4

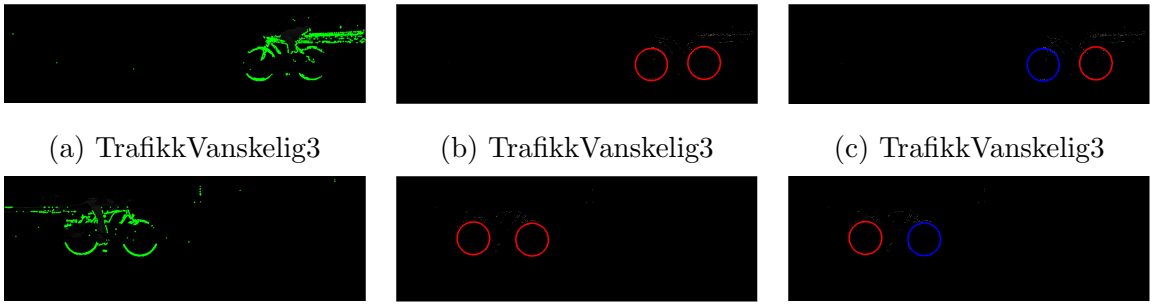
(m) TrafikkSykkel5 (n) TrafikkSykkel5 (o) TrafikkSykkel5

(p) TrafikkSykkel6 (q) TrafikkSykkel6 (r) TrafikkSykkel6

(s) TrafikkVanskelig1 (t) TrafikkVanskelig1 (u) TrafikkVanskelig1

(v) TrafikkVanskelig2 (w) TrafikkVanskelig2 (x) TrafikkVanskelig2

Figur 5.10: Utklipp av interessant område.

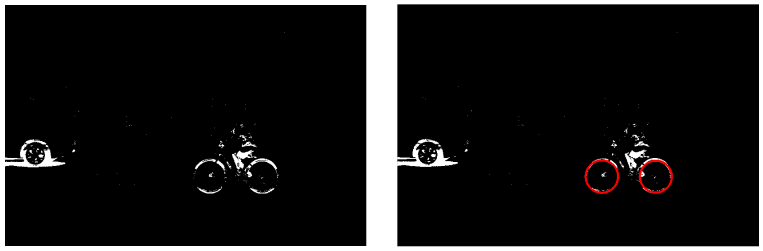


(a) TrafikkVanskelig3 (b) TrafikkVanskelig3 (c) TrafikkVanskelig3

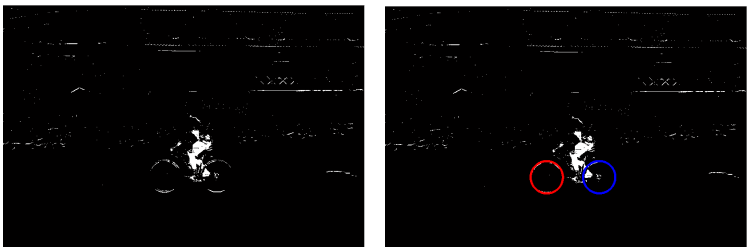
(d) TrafikkVanskelig4 (e) TrafikkVanskelig4 (f) TrafikkVanskelig4

Figur 5.11: Utklipp av interessantområde - 2.

C4 - Hough-transformen testet på binære bilder



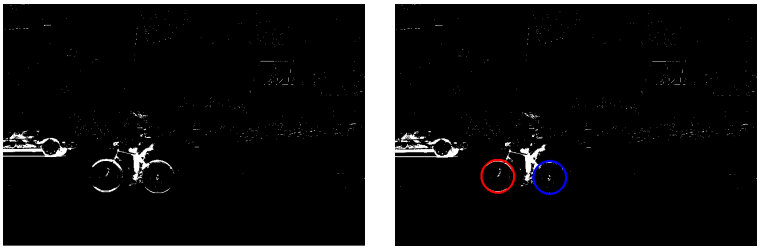
(a) TrafikkSykkel1 (b) TrafikkSykkel1



(c) TrafikkSykkel2 (d) TrafikkSykkel2

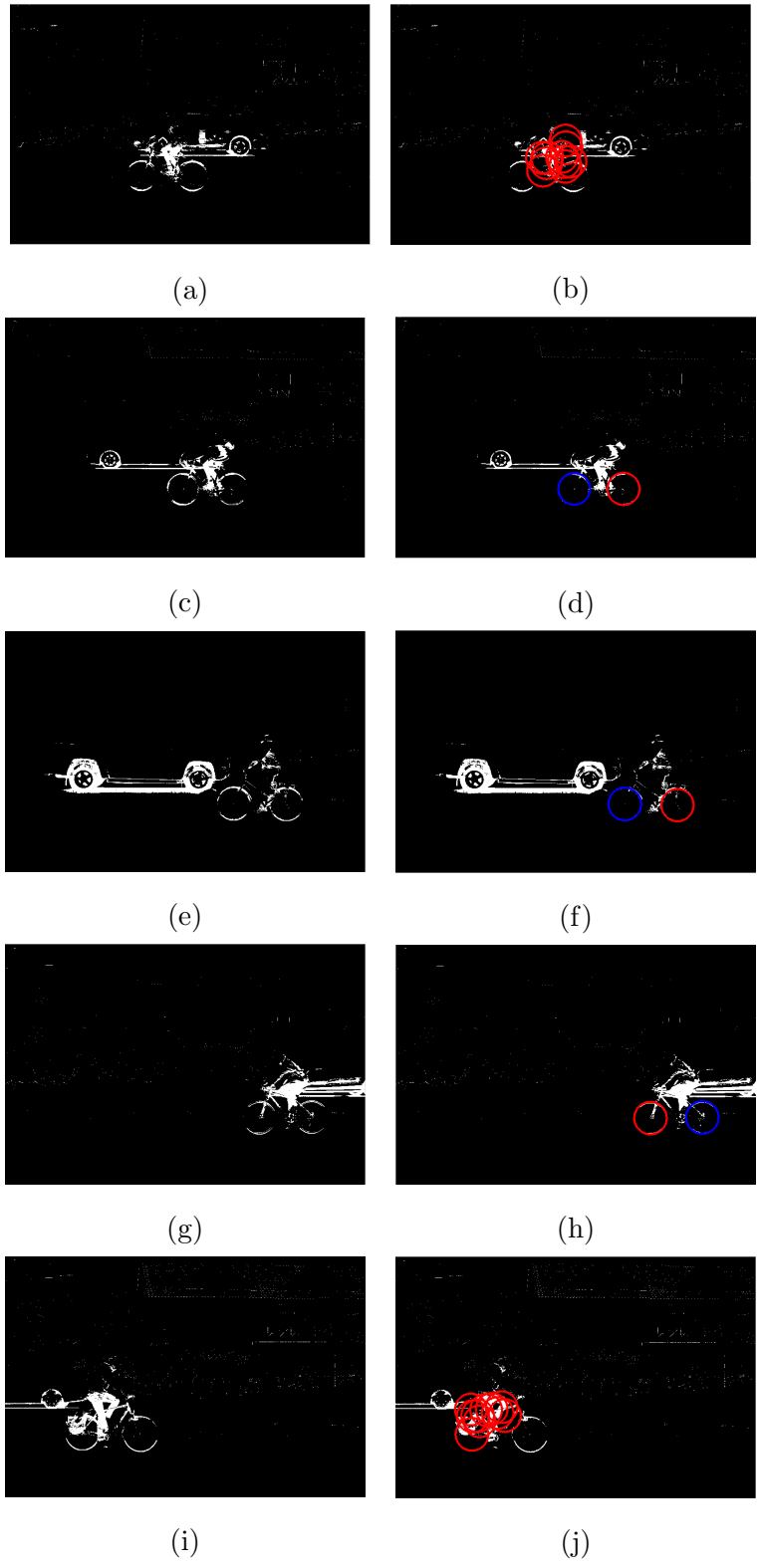
(e) TrafikkSykkel3 (f) TrafikkSykkel3

(g) TrafikkSykkel4 (h) TrafikkSykkel4



(i) TrafikkSykkel5 (j) TrafikkSykkel5

Figur 5.12: Binære bilder testet med Hough-transformen.



Figur 5.13: Binære bilder testet med Hough-transformen - 2.

D - Kildekode

Kildekoden er plassert i zipfilen. En oversikt over funksjonene er gitt her:

- Bakgrunnssub_ enkel.m
- Bakgrunnssub_ trafikk.m
- circle_ hough.m
- circle_ houghpeaks.m
- circlepoints.m
- Susan_ kant.m
- test_ enkleFigurer.m
- test_ SusanKant.m
- testBrisk.m
- testSurf.m

E - minnepenn

De originale bildene tatt med speilrefleks kameraet er lagt til på en minnepenn som leveres inn.

- Bakgrunnsbilder
- Sykkelbilder