Gjenkjenning av norske fartsskilt

Jørgen Walberg Bakke

18. mai 2004

Forord

Dette er en hovedfagsoppgave i mikroelektronikk ved institutt for informatikk på Universitetet i Oslo. Oppgaven var ferdig i mai 2004.

Prosessen har vært en utfordring på flere måter. Jeg overtok et program som var skrevet i programmeringsspråk jeg ikke hadde kjennskap til fra før. Koden til programmet var lang og krevende å sette seg inn i. Forbedringen av selve systemet, med å finne gode algoritmer som kunne forbedre gjenkjenningen og implementasjon på Field Programmable Gate Array har vært tidkrevende og lærerike prosesser. Jeg føler å ha tilegnet meg kunnskaper om flere ting gjennom denne oppgaven. Jeg har lært mye om analyse av fargebilder, og hvordan det er mulig å komme frem til gode resultater. Jeg har lært mye om å jobbe selvstendig, og utviklet meg mye med tanke på å innhente relevant informasjon og prøve ut ulike algoritmer.

Jeg vil gjerne takke min veileder, 1.amanuensis Jim Tørresen. Vi har hatt et nært og godt samarbeid. Tørresen har vært med på å drive prosessen videre med fornuftige og nyttige diskusjoner. Jeg vil også rette en spesiell takk til kjæresten min, Marianne Sælid. Hun har alltid vært der og hjulpet meg med humøret når det har gått litt trått.

Medstudenter og venner har hatt stor betydning både for oppgaven og som støttespillere. Alle disse fortjener takk, og spesielt vil jeg takke Amund H. Basmo, Håvard K. Riis, Johan I. Sæbø, Espen H. Lian, Kristian Holm, Kjetil E. Vistnes og Annika Rigenholt.

Sist men ikke minst vil jeg takke hele familien, som alltid er der. Spesielt mamma og pappa som har vært delaktig i hele prosessen.

Sammendrag

I denne oppgaven blir det beskrevet et system som er laget for å gjenkjenne norske fartsskilt i sanntid. Systemet har fått navnet Image Detection of Speed Limits (IDSL). En førsteutgave av IDSL ble utviklet i 2001 av Lukas Sekanina i samarbeid med Jim Tørresen. I denne oppgaven videreutvikles IDSL for å få det raskere og mer robust. Systemet analyserer stillbilder. Ved å analysere bildene, blir det avgjort om disse inneholder fartsskilt, og eventuelt hvilken fartsgrense skiltet informerer om.

Det første som gjøres i prosessen er å lokalisere mulige skilt i bildet. Dette gjøres ved å lete etter den røde sirkelen som omkranser fartsskiltet. For å finne sirkelen blir alle de røde pikslene i bildet lokalisert (fargefiltrering). Det blir så sjekket om noen av de røde pikslene danner en sirkel. Dette gjøres ved å bruke templater. Disse har størrelser tilsvarende det en rød sirkel i bildet kan ha. Hvis det blir funnet en slik sirkel, vil innholdet av denne bli analysert. Ut fra denne analysen blir fartsgrensen skiltet informerer om funnet.

Den første versjonen av IDSL hadde store svakheter. IDSL er derfor forbedret, slik at det er mer robust og tidsforbruket er betraktelig redusert. Forandringene som er gjort med den originale versjonen er grundig beskrevet i denne oppgaven. Det er også beskrevet hvorfor de forskjellige forandringene er gjort, og hvilke konsekvenser dette har for gjenkjenningen.

Som nevnt skal systemet gjenkjenne bilder i sanntid. Dette fører til at gjenkjenningen av hver enkelt bilde må gå så fort som mulig. Det er derfor valgt å gjøre fargefiltreringen av bildet på en Field Programmable Gate Array (FPGA). Det viser seg at det er mulig å sjekke om ca 12 bilder inneholder skilt, og eventuelt gjenkjenne fartsgrensen, hvert sekund.

Innhold

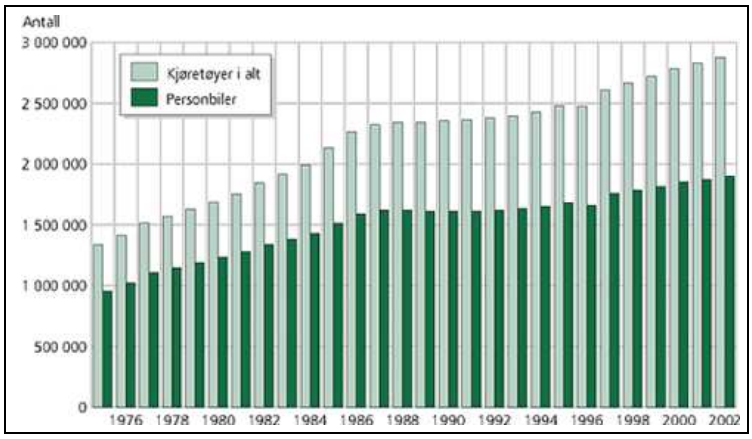
Kapittel 1

Bakgrunn

1.1 Innledning

Etter at det første kjøretøyet rullet ut på veien, har det skjedd en enorm utvikling. Det første kjøretøyet hadde halm i dekkene og kjørte på humpete grusveier. Den gangen var det kun de velstående i samfunnet som hadde økonomisk evne til å eie et kjøretøy. Dagens bil er motorisert, har støtdempere, kjøres stort sett på jevn asfalt og veistandarden er vesentlig bedret. Dette har gjort bilen til et meget nyttig og behagelig fremkomstmiddel. Bilen har i moderne tid blitt allemannseie, noe som har ført til at antall biler har vokst dramatisk og antallet øker fortsatt for hvert år som går. Ifølge figur 1.1, var det i 2002 nesten 2 000 000 personbiler og til sammen nesten 3 000 000 motoriserte kjøretøy i Norge [22].

Motorene som sitter i dagens biler er betydelig kraftigere enn tidligere, noe som medfører at de går vesentlig raskere. Dette sammen med den økede trafikktettheten og informasjonsmengden (antall skilt o.l), har ført til at det er mer krevende å være sjåfør nå enn tidligere. Antall ulykker, og alvorlighetsgraden av disse, har hatt en jevn stigning siden bilen ble introdusert. Det er derfor nå økende fokus på trafikksikkerhet. I mange tilfeller kan årsaken til ulykkene føres tilbake til føreren, noe som har ført til en sterk fokusering på tekniske løsninger som kan hjelpe førerne. Enkelte av disse kan til og med hindre at førerne begår feil. De første bilene var åpne vogner uten sikkerhetsutstyr. Nåtidens biler er spekket med utstyr som skal sikre de som sitter inne i dem. Trepunkts sikkerhetssele sikrer at personer ikke blir slengt ut av bilen og at de påføres minst mulig skade ved et eventuelt sammenstøt. Det er stålbjelker



Figur 1.1: Oversikt over antall kjøretøyer i Norge [22]

i dørene og kollisjonsputer både ved fører- og passasjersetet i alle nyere biler. ABS (anti blocking system) gjør det lettere å ha kontroll over bilen under kraftig nedbremsing. Grunnet kontinuerlig satsing på sikkerhet, blir bilene stadig bedre og antallet alvorlige ulykker har stagnert de siste årene [22].

1.1.1 Systemer under utvikling

I [10] blir det beskrevet hva som kan ventes av utstyr i biler i 2020. Det blir fortalt om førerovervåkingssystemer, kollisjonsvarsling, ruslås og head up display (HUD).

Førerovervåkingssystemer skal hindre at bilen havner i motgående kjørefelt eller kjører av veien. Nissan har allerede plassert et slikt system i en av sine modeller. Et tilsvarende system er også under utvikling i Australia [7]. ACC (Adaptive Cruise Control) vil hjelpe føreren til å avpasse farten etter trafikkforholdene [11]. Det vil si at man avpasser farten slik at avstanden til bilen som kjører foran er forsvarlig. Flere bilmerker har ACC installert i sine toppmodeller.

Kollisjonsvarsling er et system som skal hjelpe til med å analysere trafikkbildet. Det består av sensorer som rettes mot ethvert objekt som kan komme i konflikt med bilen, og informerer føreren om kritiske objekter. Et system som overvåker *føreren*, skal være med på å forhindre at vedkommende sovner bak rattet. Dette systemet kan også forhindre at førere kjører under påvirkning av alkohol eller narkotika ved å analysere

øyne og ansiktsuttrykk.

Et system som kombinerer kollisjonsvarsling og føreroverv˚aking er under utvikling i Australia [7]. Dette systemet følger med på hvor føreren ser på veien, og varsler bare om farer som føreren ikke har sett direkte på.

Ruslås kan også med tiden bli standard i fremtidens biler. Dette er et måleinstrument der føreren må avgi utåndingsprøve før bilen kan startes. Dette instrumentet er allerede påbudt innen offentlig transport i flere svenske kommuner.

Informasjonen føreren normalt forholder seg til ved hjelp av instrumenter inne i bilen, er begrenset til informasjon om bilens tilstand. Den eneste informasjonen det er vanlig å få om omgivelsene, er trafikkmeldinger på radioen og eventuelt utetemperatur med varsling om mulig fare for glatte veier. Det ville være en fordel for føreren om det ble gitt relevant informasjon om strekningen bilen befinner seg på. På denne måten vil føreren ikke aktivt trenge å oppsøke informasjonen og selv bedømme hva som var relevant.

HUD (Head-Up-Display) vil kunne gi føreren trafikkinformasjon direkte på et lite område på frontruta. Plasseringen på frontruta vil være slik at den ikke er i veien for førerens utsyn. Tanken er å informere om kødannelser, friksjonsforhold, ulykker, hastighet mm. uten at føreren behøver å flytte blikket fra veien. Det vil også kunne bli gitt opplysninger til føreren om hvilken fartsgrense det er på den aktuelle strekningen.

Det er nettopp et slikt system, som kan muliggjøre visning av fartsgrense, denne oppgaven beskriver. Systemet har fått navnet Image Detection of Speed Limits (IDSL). Dette systemet gjenkjenne fartsskilt ut fra bilder tatt av et kamera plassert i bilen. Vi har vel alle vært litt usikre på hvilken fartsgrense det er på den strekningen vi kjører på. I slike situasjoner vil IDSL være til hjelp.

IDSL er et system som skal plasseres i bilen. Det vil bestå av et kamera som filmer veien og noe av omgivelsene, selve “hjernen” som skal gjenkjenne de fartsskiltene som blir filmet og et panel som viser hvilken fartsgrense det er på veistrekningen.

Videre i dette kapittelet skal jeg belyse hovedgrunnen til å ha IDSL i biler, samt drøfte hvilke fordeler bilister kan ha av et slikt system. Til slutt i dette kapittelet blir ulike systemer for gjenkjenning av skilt introdusert.

1.2 Problemstilling

Denne oppgaven er en del av et større prosjekt som skal utvikle et system som skal gjenkjenne og tolke fartsskilt. Det er to hovedmål med denne oppgaven. Det ene er å forbedre og videreutvikle den opprinnelige versjonen [21] for å gjøre gjenkjenningen mer robust. Det andre er å gjøre rede for hvordan det er mulig å lage en prototyp som klarer å analysere minst 10 bilder i sekundet.

I en eventuelt kommersiell versjon av IDSL vil det være nødvendig å benytte en billig prosessor. Det vil si en prosessor som er vesentlig tregere en prosessoren IDSL testes på. Tidsforbruket for gjenkjenning av et bilde vil gå opp i forhold til tidene som beskrives i denne oppgaven. 10 bilder er satt som grense, for at IDSL skal være kapapel til å gjenkjenne flere bilder per sekund også med den billige prosessoren.

1.3 Motivasjon

Hovedgrunnen til at det er ønskelig å plassere IDSL i bil er helt klart å øke trafikksikkerheten. Dersom bilister blir informert om at fartsgrensen overskrides, vil det sannsynligvis føre til at færre kjører fortere enn det som er tillatt. Det skal ikke mye fantasi til for å forstå at ulykkene vil bli mindre alvorlige dersom biler holder fartsgrensen framfor å kjøre for fort. Det vil også være lettere å stoppe dersom farten er redusert, noe som kan forhindre ulykker. Dette er spesielt viktig i de områder hvor det er fotgjengere og andre myke trafikanter. Dette er en gruppe som blir påført store skader i ulykker der biler er involvert. Det er åpenbart at høy fart vil øke sannsynligheten for at ulykker ender med tragisk utfall og store skader, eller i verste fall død. I [2] er det beskrevet at antall ulykker med personskader vil gå ned med 20% og ulykker med alvorlig personskader vil gå ned med 37%, dersom bilene ikke har mulighet til å overskride fartsgrensen. Ved å kombinere et fartsbegrensende system med informasjon om vær-, trafikk- og friksjonsforhold, mener [2] at ulykkestallene kan senkes med henholdsvis 36% og 59%. Dette vil ha meget positiv effekt for den enkelte og for samfunnet, og er helt klart noe som bør etterstrebes.

IDSL kan også hjelpe førere som har *oversett* skilt, til å holde riktig fart. Det kan også tenkes at IDSL vil kunne si fra om man kjører vesentlig *under* fartsgrensen, og da få føreren til å *øke* farten opp mot denne. Dette kan føre til at antall farlige forbikjøringer går ned. Bilister bak en bil

som kjører langt under fartsgrensen kan fort bli frustrerte og irriterte, og vedkommende kan da være en fare i trafikken.

Med IDSL installert i bilen slik at hastigheten automatisk kan begrenses, kan behovet for fysiske veihindringer som fartsdumper reduseres [24]. Da vil ikke bilen kunne kjøre raskere enn det veistrekningen er beregnet for, noe som kan føre til at slike hindringer er overflødige. Det betyr at penger som i dag benyttes på fysiske veihindringer i stedet vil kunne bli benyttet til å forbedre veiene eller til andre formål.

1.4 Bruksområder

IDSL er så langt basert på norske fartsskilt og dermed anvendelig på biler som kjører på *norske* veier. Den mest grunnleggende måten å bruke dette systemet på, vil være å plassere et panel der fartsgrensen vises inne i bilen. En variant kan være å koble systemet til speedometeret som sitter i bilen. Dermed kan man med fargekoder gi føreren av bilen beskjed om hvordan farten er i forhold til fartsgrensen. Panelet kan bytte farge til f.eks rødt, dersom man kjører fortere enn fartsgrensen. Om man kjører vesentlig under fartsgrensen kan panelet bytte farge til f.eks gult.

Det vil også være mulig å informere føreren på en annen måte. Det kan komme lydalarm inne i kupeen dersom fartsgrensen overskrides. Dette kan virke irriterende på føreren, så lyden bør ikke komme før fartsgrensen er overskredet over et vist tidsrom. Lyden kan ha flere nivåer, begynne rolig når fartsgrensen overskrides, for så å bli sterkere etter hvert. Ulempen med denne alarmen er at føreren kan bli forstyrret, og dermed havne i farlige situasjoner.

En videreutvikling vil være at man ikke kun nøyer seg med å opplyse føreren at hastighet er høyere enn den tillatte på den aktuelle veistrekningen ,men også fysisk går inn og begrense hastigheten. IDSL kan da kobles til gasspedalen, slik som det er gjort i [24]. Her er det brukt noe som kalles for “Active Accelerator” (AA). Det vil si at gasspedalen blir tyngre når fartsgrensen overskrides. Det fører til at man må gjøre noe aktivt for å overskride fartsgrensen. Dersom det oppstår en farlig situasjon hvor fartsgrensen må overskrides for å unngå fare, kan AA kobles ut ved å tråkke pedalen helt inn. Eksempel på en slik situasjon kan være at den kalkulerte strekningen ved forbikjøring ikke er tilstrekkelig for å komme inn foran bilen som passeres. Dersom man kjører vesentlig *saktere* enn fartsgrensen, kan pedal motstanden bli mindre, slik at dette er en indikasjon på at føreren bør sette opp farten.

Et annet mulig anvendelsesområde vil være å koble IDSL direkte til "cruise controllen". Dette vil føre til at bilen automatisk vil følge fartsgrensen. Denne anvendelsen vil føre til at sjåføren konsentrerer seg mer om trafikken, i og med at man ikke er nødt til å følge så nøye med på fartsskilt og speedometeret. På den annen side kan sjåførene føle at de ikke trenger å følge så godt med, fordi de stoler på at bilen styrer farten automatisk. Dersom man lar tankene fly eller lar seg distrahere av andre passasjerer vil det lett kunne oppstå farlige situasjoner, og i verste fall ulykker. Et annet faremoment kan oppstå dersom IDSL av en eller annen grunn ikke finner *riktig* fartsgrense. Dette kan skje, f.eks ved at noen har klusset på et 30-skilt, slik at det ser ut som et 80-skilt. Dette kan delvis unngås ve å sjekke hvilke skilt som kan komme i følge skiltbestemmelser, som beskrives nærmere i 1.5.2. På vinterstid kan skilt være tildekket etter snøbrøyting og IDSL vil da ikke kunne identifisere overgang fra en fartsgrense til en annen, f. eks når man kommer fra landevei inn i tettbygd strøk. Disse konsekvensene gjør det nødvendig at føreren selv kan velge å ha denne funksjonen innkoblet eller ikke. Denne funksjonen vil være mest hensiktsmessig på landeveier og motorveier. Det må også være mulig å overstyre funksjonen ved å trå på gasspedalen for å øke hastigheten eller bremse for å redusere den.

1.5 Avgjørende egenskaper for pålitelighet

Når det skal lages et system som IDSL, er sikkerheten avgjørende. Det må derfor legges vekt på pålitelighet under utviklingen av systemet.

1.5.1 Sikker forandring av fartsgrense

I og med at IDSL skal gjenkjenne fartsskilt og veilede førere i henhold til hva som blir funnet, er det viktig at informasjonen som formidles er riktig. Det kan ha alvorlige konsekvenser dersom føreren får beskjed om feil fartsgrense. Systemet må derfor være helt sikker på at riktig fartsgrense er funnet før den blir videreformidlet til føreren. Dette er av avgjørende betydning for påliteligheten til systemet.

1.5.2 Pålitelighet

Dersom IDSL feiltolker informasjonen som mates inn og av den grunn oppgir feil fartsgrense, vil troverdigheten til systemet svekkes. Brukere av systemet vil etter en slik episode, aldri være helt sikker på at riktig fartsgrense vises. Derfor må det stilles strenge krav før konklusjonen om forandring av fartsgrense trekkes. Det er også viktig at kravene ikke er så strenge at fartsgrensen ikke blir byttet når fartsgrensen byttes. Systemet må ha mulighet til å angi "ukjent fartsgrense" dersom det har funnet noe som kan være fartsskilt, men ikke er sikker på om det faktisk er det. Dette bør også benyttes dersom IDSL ikke er i stand til å foreta en sikker identifisering av fartsgrensen på skiltet. I slike tilfeller må føreren gjøres oppmerksom på at IDSL ikke lenger kan "ta ansvaret for" riktig fartsgrense og overlate det til føreren. Dette bør inntreffe så sjeldent som mulig, men det er bedre at systemet gir beskjed om at det ikke er i stand til å foreta en sikker identifikasjon enn å vise feil fartsgrense.

Det er flere egenskaper som kan være med på å gjøre gjenkjenningen sikrere. Det er skiltbestemmelser som regulerer hvilke skilt som kan etterfølge hverandre. Dette betyr at 80 skilt aldri vil bli etterfulgt av et 30 skilt. Dette kan være med på å eliminere alternativer for fartsgrenser. I Norge er det skilt på begge sider av veien dersom fartsgrensen på den aktuelle strekningen skal forandres. Dette er også informasjon som systemet kan benytte for å være sikker på at fartsgrensen skal forandres. Disse kriteriene blir nærmere beskrevet i avsnitt 6.4.

I en utviklingsprototyp vil det være ønskelig å lagre tilfeller der systemet er usikker på hva fartsgrensen er. Dette kan hjelpe til med å gjøre gjenkjenningsalgoritmen bedre.

1.6 Utfordringer for skiltgjenkjenningssystem

Det er mange utfordringer når det skal lages et system som skal finne skilt som befinner seg langs bilveier. I motsetning til innendørs, hvor man har konstant belysning, er det ikke mulig å styre lysforholdene utendørs. Lys og skygge gjør gjenkjenningen av skilt til en krevende oppgave [6, 28]. Det er store forskjeller på lysnivået fra tidlig på dagen til kvelden, noe som påvirker hvordan skilt blir gjengitt når det blir tatt bilde av dem. Skygge vil også spille inn på gjengivelsen. Det vil være stor forskjell på hvordan skilt fremstår om de står i skygge eller ikke.

I tillegg til lys og skygge vil hindringer, forurensning og langvarig eksponering av sollys være med på å gjøre skiltgjenkjenning til en komplisert oppgave [6]. Langvarig eksponering av sollys vil føre til at fargene på skiltet falmer, mens forurensning vil gjøre skilt skitne. Begge disse faktorene vil gjøre at skilt ikke blir like tydelige som de burde være. Slike hensyn må bli tatt når systemet utvikles. At objekter dekker skilt er umulig å unngå. Noen hindringer kan elimineres, ikke av selve systemet, men av de som har drift- og vedlikeholdsansvar for skiltene. Eksempel på slike hindringer kan være greiner som henger foran skilt. Hindringer som lyktestolper, bygninger og ikke minst andre kjøretøyer er det ikke mulig å eliminere. Dette kan føre til at skilt blir helt eller delvis skjult. Systemet bør imidlertid kunne gjenkjenne skilt som er delvis tildekket. Et skilt vil ellers forhåpentligvis ikke være tildekket på alle bildene som tas fra en bil i fart.

For norske forhold er det klart at snø kan være med på å gjøre gjenkjenningen meget krevende, og til tider umulig. Snø kan legge seg på deler eller hele skiltet. Dette kan også føre til at mennesker har problemer med å tyde skiltet. Det kan ikke forventes at systemet skal gjenkjenne skilt i disse situasjonene, så føreren må selv avpasse farten etter forholdene.

Hærverk på skilt kan også føre til problemer for gjenkjenningen. Det er spesielt to typer hærverk som forekommer. Tilgrising med maling og hull eller skader fra våpen eller stein. Dette kan forandre karakteristikken til skiltet betraktelig som kan være utslagsgivende for gjenkjenningen.

1.7 Andre systemer

Nedenfor gis det en oversikt med forklaring av enkelte systemer som er laget for å bedre trafikksikkerheten. Det er kun de systemene som tar for seg bilers fart og gjenkjenning av skilt som blir beskrevet og forklart.

1.7.1 ISA

Intelligent speed adaptation (ISA) er et system som forhindrer at biler kjører fortere enn de har lov til [24]. Dette er et system som har vært prøvet ut i fire svenske byer (Borlänge, Lindköping, Lund og Umeå) i perioden 1999 til 2002. Hovedmålet med prosjektet er å forhindre at biler kjører for fort i tettbygde strøk.

ISA-systemet bruker Global Positioning System (GPS) som mottar informasjon om hvor bilen befinner seg. Denne informasjonen blir så sjekket mot et digitalt kart som befinner seg i bilen. Kartet inneholder informasjon om fartsgrensen i de aktuelle områdene. Gjeldende fartsgrense på den aktuelle strekningen, blir vist på display inne i bilen. Dersom bilen holder en hastighet som er høyere enn det som er tillatt på veistrekningen, vil "active accelerator" (AA) slå inn. AA er forklart i avsnitt 1.4, og føreren får beskjed om at farten er for høy.

Kartet i ISA systemet begrenser seg til sentrale deler av byen. Det er mulig å stille inn ISA manuelt på en fartsgrense som ikke skal overskrides utenfor områdene kartet dekker. Målinger som er foretatt viser at de bilene som har dette systemet installert ikke øker reisetiden til tross for at bilen ikke overskrider fartsgrensen. Dette forklares med at kjøringen blir jevnere med færre oppbremsinger og stopp, når man holder fartsgrensen.

Det var flere av testdeltakerne som i løpet av testperioden hadde en følelse av at systemet overvåket dem. Det er lett å forstå at dette er en negativ opplevelse som flere kan bruke som argument mot å ta systemet i bruk. Prisen er en ulempe med ISA-systemet. Installasjon av et slikt system i en bil som ikke allerede har en GPS vil koste omtrent 20 000 SEK. Dette vil medføre store kostnader for alle som skal ha ISA systemet. Det vil også være avgjørende at man har detaljerte digitale kart som inneholder fartsgrensene i større geografiske områder. Oppdatering av kartene som sitter i bilene vil også være en utfordring. Alle bilene må få nye kart for å være oppdatert med nye veier, eller endringer i fartsgrenser. Det vil si at systemet vil kreve mye vedlikehold og vil være kostbart i anskaffelse og oppdatering.

1.7.2 Sanntids gjenkjenning av veiskilt basert på genetisk algoritme

Et system som identifiserer fartsskilt i sanntid, er beskrevet i sin helhet i [14]. Systemet benytter et CCD (Charge-coupled device) kamera som er plassert i en bil. Dette sender bilder til en datamaskin som analyserer dem. Bildene har en størrelse på 320x240 piksler. Systemet klarer å sjekke 5 bilder i sekundet. For å finne skilt benyttes en kombinasjon av kantdeteksjon og fargefiltrering. Fargefiltreringen kalles Simple Vector Filter (SVF). Metoden tar utgangspunkt i fargebildet og fjerner alt som er fargeløst. Til kantdeteksjon benyttes 3x3 Laplace filter og resultatet er et bilde hvor uinteressant informasjon er fjernet. Det som ikke er filtrert

bort bearbeides videre.

Når en sirkel i det filtrerte bildet skal lokaliseres, gjøres søket bare på en mindre del av bildet. Området som sjekkes er 256x128 piksler stort, og befinner seg oppe til høyre i bildet. Dette området er valgt fordi det erfaringsmessig er størst sannsynlighet for at skiltet befinner seg der. For å lokalisere sirkler benyttes genetisk algoritme (GA). Metoden går ut på at objekter i området beskrives ved hjelp av vektorer. Disse blir sammenliknet (ved hjelp av GA) med en vektor som beskriver en ideell sirkel. Den vektoren som likner mest på den ideelle sirkelen blir ansett for å være et skilt.

Det er ikke laget noen metode for å analysere innholdet av sirkelen. Det er heller ikke gitt noen oppsummering av resultatene. Det er vanskelig å si noe om hvor godt metoden egner seg. I [14] går det frem at alle bildene som er testet er tatt på dagtid og i godt vær. Dette betyr antakelig at systemet så langt ikke er robust nok til å kunne benyttes i stor skala. Resultater av tester viser at GA ikke alltid finner skilt, noe som forklares med at bilder tatt fra bil ofte har for dårlig kvalitet.

1.7.3 Sanntids skiltgjenkjenning som benytter aktivt kamera

I [17] blir det beskrevet et system som finner og gjenkjenner farts- og retningsskilt. Systemet benytter to kameraer, ett med vidvinkel som finner kandidater, og ett med telelinse som rettes mot kandidatene. For å finne kandidater til skilt blir det gjort fargefiltrering. For at minst mulig annet enn skilt skal komme ut av filtreringen, benyttes informasjon om skiltenes egenskaper. Det søkes derfor etter hvitt, som må ha røde eller blå naboer for å være del av skilttypene. Under fargefiltrering kan lysforhold virke inn på filtreringsresultatet slik at det ønskede resultat ikke oppnås. For å motvirke lysets innflytelse blir filtrering gjort flere ganger med forskjellige krav for fargene. Den filtreringen som gir det beste resultatet blir brukt videre.

For å finne ut om det er skilt i det filtrerte bildet blir det brukt informasjon om formen på skiltene. Det blir gjort kantdeteksjon på det filtrerte bildet. For å finne runde skilt benyttes Hough-transformasjon, som beskrives i avsnitt 2.3.1. Den går ut på å finne gradienten (90° på ethvert punkt i sirkelen) til alle kantene. Dersom mange gradienter går gjennom et og samme punkt, ansees det for å være sentrum av skiltet. For rektangulære skilt sjekkes horisontale og vertikale kanter. Finnes det parallelle linjer både vertikalt og horisontalt med omtrent like stor avstand, ansees

det for å være kantene på retningsskiltet.

Telelinsen blir så rettet mot kandidat til skilt. For runde skilt blir kandidaten normalisert før templater av forskjellige fartsskilt testes på kandidaten. Dette vil si at kandidaten blir gjort om til en gitt størrelse og sammenliknet med fartsskilt som har samme størrelse. Kandidaten anses for å være det fartsskiltet som passer best overens med templatet. For retningsskilt blir piler og tekst skilt fra hverandre. Teksten blir gjenkjent ved å sjekke den opp mot templater. Retningen på pilene blir avgjort ved å sjekke hvor det er høyest konsentrasjon av hvite piksler i et histogram.

I følge [17] blir 97% av fartsskiltene, og 100% av retningsskiltene lokalisert. Metoden er ikke like treffsikker når det gjelder identifikasjonen av skiltet. Resultatene her er henholdsvis 46% og 23%. Det er også avdekket problemer med telefotokameraet. Problemet er at det ikke er utviklet en metode for å få kameraet til å bevege seg til kandidaten så raskt som nødvendig for å kunne få en sikker identifikasjon.

1.7.4 Ny tilnærming for sanntids gjenkjenning av trafikkskilt

Det introduseres en alternativ metode for gjenkjenning av fartsskilt i [20]. Den går ut på å sette opp ekstra skilt i tillegg til det eksisterende skiltet. Det nye skiltet er spesialdesignet for elektronisk visuell gjenkjenning. Det nye skiltet som er foreslått satt opp er firedelt i vertikal retning. Hver av kolonnene har hver sin farge, og sammensetningen av fargene representerer de forskjellige skiltene.

Som [6] kommenterer, vil dette være en svært ressurs- og tidkrevende måte å gjenkjenne skilt på. Grunnen er at det må settes opp ekstra skilt i tillegg til de eksisterende. Det vil også være dyrt å vedlikeholde, siden disse skiltene må byttes ut like ofte som vanlige skilt.

1.7.5 Gjenkjenning av trafikkskilt ved hjelp av nevrale nett

For å finne farts-, parkering forbudt- og stoppskilt, benyttes to nevrale nett sammen med Laplacian av Gaussian filter (LOG filter) [19]. Først benyttes LOG filter sammen med et analyseverktøy. Dette reduserer informasjonen i bildet til nesten en tredjedel. Dette bildet blir så brukt som input til nevralt nett som benyttes som fargefilter. Dette filtrerte bildet blir så delt opp i mindre deler, som blir normalisert og brukt som

input i et nytt nevralt nett. Det nevrale nettet sjekker om bildedelen tilfredsstiller kravene til skilt. Dersom det ikke gjør det blir neste kandidat sjekket.

Dette systemet får veldig gode resultater, over 95% gjenkjenning på de aller fleste skilt. For å teste systemet er det valgt fire forskjellige strekninger. Systemet er testet på disse strekningene på to forskjellige tidspunkter på to forskjellige dager. Det er ikke sagt noe om hvordan værforholdene på de to dagene var. Det er dermed umulig å si hvor robust dette systemet egentlig er.

Et alternativ som flere har foreslått for meg er å sette opp sendere som sender informasjon til biler som passerer at fartsgrensen er forandret. Et annet alternativ er å ha passive elektroniske "markører" ved skilt som reflekterer signaler biler sender ut. Et slikt system benyttes i bomringer.

1.8 Oversikt over resten av rapporten

Som nevnt tidligere dreier denne oppgaven seg om å utvikle et system som skal lokalisere og gjenkjenne norske fartsskilt (kalt IDSL). Ideen og oppbygningen er gjort av Lukas Sekanina i samarbeid med Jim Tørresen, og er beskrevet i [21]. IDSL finner skilt ved å analysere bilder. For å finne skilt i bildet, lokaliseres den røde sirkelen som omkranser norske fartsskilt. Dersom denne sirkelen blir funnet, blir innholdet tydet og fartsgrensen fastlagt. Jeg har videreutviklet IDSL med fokus på tidsforbruk og robusthet. Det er lagt inn en tidtaker som måler tiden på de forskjellige fasene av gjenkjenningen.

Teorien bak de metodene som blir benyttet i IDSL blir forklart i kapittel 2. En begrunnelse for de valg som er gjort blir også gitt. I kapittel 3 er det en beskrivelse av hvordan IDSL fungerer og hvordan systemet kommer frem til hvilket tall som står inne i fartsgrenseskiltet. I kapittel 4 blir det gjort rede for de forandringene jeg har gjort med IDSL og hvordan disse påvirker gjenkjenningen. Implementering på Field Programmable Gate Array (FPGA) blir forklart i kapittel 5, og i kapittel 6 blir det skissert mulige forbedringer og videre utvikling av IDSL. Kapittel 7 inneholder konklusjon.

Kapittel 2

Teori om bildebehandling

I dette kapittelet blir teorien bak enkelte metoder som er utviklet for å behandle bilder beskrevet. Det er foretatt vurderinger av styrker og svakheter ved de ulike metodene. Det blir også gitt begrunnelse for de valg som er tatt for å avgjøre hvilken metode som skal benyttes.

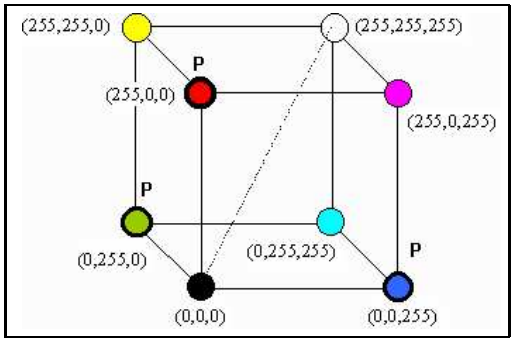
En standart teknikk for å lokalisere og tyde veiskilt består av tre steg [13]. Farge segmentering eller farge filtrering gjøres først, for å redusere områder hvor det skal søkes etter skilt. I steg to benyttes templater for å lokalisere skilt. Det siste steget gjenkjenner innholdet på skiltet ved hjelp av templater eller nevrale nett.

2.1 Fargerepresentasjon

For en applikasjonstype som IDSL er det helt avgjørende at måten farge representeres på tar vare på informasjon i bildet. Det er også viktig at det er mulig å hente ut denne informasjonen på en sikker, effektiv og entydig måte. Fargerepresentasjon baserer seg på Thomas Youngs klasseteori (1802) om at alle farger kan reproduseres ved å blande tre primærfarger [1]. I den digitale verden finnes flere måter å representere farger på og her vil enkelte av disse bli beskrevet nærmere [13, 23]. Det er valgt å legge vekt på de fargerepresentasjonsmåter som er mest aktuelle for applikasjoner av denne typen.

2.1.1 RGB

RGB (Red, Green, Blue) er den vanligste måten å beskrive farger digitalt på. RGB har en separat verdi for hver farge, som vil si at det trengs tre verdier for å beskrive fargen på et piksel. Verdiene blir beregnet etter hvor stor mengde det er av de forskjellige fargene i det bestemte området pikselet representerer. En grafisk fremstilling er vist i figur 2.1, hvor fargene vist med P er primærfarger. Det er mange systemer som benytter RGB på en mer eller mindre direkte måte for å finne skilt [14, 18, 19, 21].

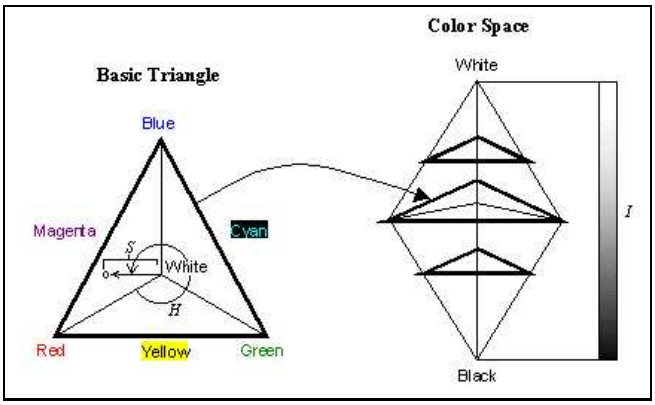


Figur 2.1: Grafisk fremstilling av RGB [8]

2.1.2 HSI

Lysforhold har stor innvirkning på hva RGB verdiene blir. Derfor er det utviklet flere representasjonsmåter som motvirker denne effekten. Den beste av disse er HSI (Hue, Saturation, Intensity) [13]. HSI har mange likhetstrekk med den måten mennesker oppfatter farger på. Hue (H) er verdien for den rene fargen som blir gitt i grader. Gradene er en omskriving av bølgelengden, som er den måten mennesker oppfatter farge på. Mennesker kan se farger som har bølgelenge 400-700 nm. Rød er gitt verdien 0 grader, grønn er gitt 120 grader og blå er gitt 240 grader.

Saturation (S) beskriver mengden eller renheten av fargen. Rød og rosa har lik H-verdi, men det er S-verdien som skiller disse fargene fra hverandre. Intensity (I) beskriver intensiteten på lyset. En grafisk fremstilling av HSI er vist i figur 2.2, hvor sammenhengen mellom de forskjellige verdiene og fargene vises. Det er utviklet varianter av HSI, som f.eks HSV og HLS. Det som skiller disse fra hverandre er at intensity (I) er byttet ut med en alternativ måte å beregne lysintensiteten på. Det er flere skiltgjenkjenningsapplikasjoner som har valgt å benytte HSI [6, 28].



Figur 2.2: Grafisk fremstilling av HSI [8]

2.1.3 YUV

YUV ble utviklet da TV gikk over fra å sende bare i svart/hvit, til også å være fargesendinger. TV-signaler blir sendt på formen YUV, som gjør dem kompatible med både svart/hvit og farge TV-er. Verdiene til YUV er en omregning av RGB verdiene.

Y kanalen er en svart/hvit representasjon av selve bildet og svart/hvit TV-er benytter bare denne verdien. U og V er fargeinformasjon som benyttes i tillegg til Y verdien av farge TV-er. U kanalen inneholder informasjon om fargespekteret fra rødt til gult, mens V er informasjon om fargen fra gult til blått. Enkelte kameraer benytter YUV for å representere farger. YUV har også blitt benyttet for å lage system for å gjenkjenne skilt [17].

2.1.4 Valg av fargerepresentasjon

Som nevnt er RGB følsom ovenfor forandringer i lysforholdene. Følgene av forandringer er at samme farge får forskjellige verdier avhengig av styrken på lyset som treffer gjenstanden. I og med at systemet skal behandle bilder som er tatt utendørs, vil lysforholdene ikke være like hele tiden. Dette gjør det krevende å bestemme grenser mellom farger ved å benytte RGB. Måten fargene beskrives på er også ganske kompleks. Metoden kan sees på som en 3-dimensjonal graf med en fargeverdi på hver akse, noe som er vist i figur 2.1. At alle verdiene må benyttes for å karakterisere en farge, gjør det vanskelig å finne entydige skillelinjer mellom fargene. Den klart største fordelen ved bruk av denne representasjons-

formen er at det ikke er nødvendig å gjøre utregninger for å komme frem til verdiene som beskriver fargen på hvert enkelt piksel. Dette fordi de fleste kameraene benytter RGB som fargerepresentasjon

Å benytte HSI har en klar fordel ved at det bare er én verdi som beskriver farger. Dette gjør det enkelt å bestemme grenser mellom forskjellige farger. Derimot er H-verdien konstant langs gråtoneaksen. For farger som ligger nær gråtoneaksen, er H-verdien veldig ustabil [25]. En annen ulempe er at det ikke er mulig å få HSI-verdiene direkte fra video eller stillbilder. Det kreves utregning for å komme frem til verdiene. Utregningen av H-verdien er ganske krevende. Nedenfor er formelen gjengitt:

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]$$

Dersom HSI skal benyttes i dette prosjektet, blir det veldig mange regneoperasjoner for hvert enkelt bilde. Dette vil øke prosesseringstiden, noe som er uønsket. Studier har vist at H-verdien for bilder tatt utendørs, er avhengig av lysforholdene [28]. Det betyr at HSI er mindre egnet enn først antatt.

For systemer som benytter YUV er resultatene etter filtreringen gode [17]. Dette gjør YUV til kandidat som en måte å representere farge på. På den annen siden er det også mulig å komme frem til gode filtreringsresultater ved bruk av RGB [4, 13, 21]. Denne oppgaven er en del av et større prosjekt, og det er allerede laget metoder hvor filtreringsresultatene er gode [21]. Det er derfor valgt å fortsette med RGB som fargerepresentasjon i IDSL.

2.2 Segmentering

Segmentering er en av de viktigste stegene for å analysere bilder [23]. Målet med segmentering er å gi hvert piksel en etikett som sier noe om denne pikselens tilhørighet til en eller annen gruppe piksler (*region*). Dersom informasjonsmengden om et bilde som skal segmenteres er stor, øker kvaliteten på segmenteringen [23].

Det er tre hovedformer for segmentering; terskling, regionsbasert segmentering og kantbasert segmentering. Det er lite informasjon om segmentering av fargebilder, men det er mulig å overføre teori om segmentering av gråtonebilder. Dette gjøres ved å separere lagene som beskriver

bildet og benytte samme metode på hvert av lagene, eller at bildet blir gjort om til et gråtonebilde. Under kommer en kort innføring i enkelte segmenteringsmetoder.

2.2.1 Terskling

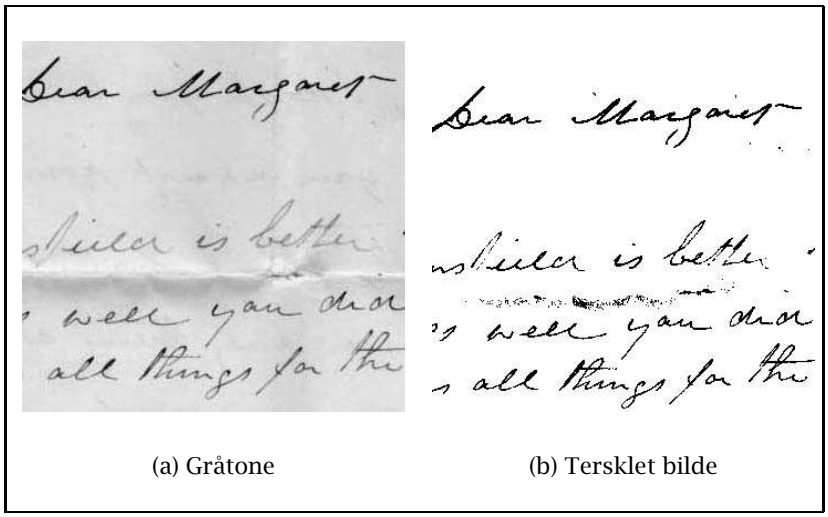
Terskling er den enkleste formen for segmentering og egner seg godt til sanntids segmentering. Terskling får frem alle de pikslene som tilfredsstiller visse krav med hensyn på gråtone eller farge.

Global terskling

Global terskling på et gråtonebilde er den enkleste måten å terskle et bilde på [9]. Global terskling deler bildet inn i forgrunn og bakgrunn, og lager et binært bilde. For å terskle et slikt gråtonebilde settes en global terskel (T). Alle pikselverdier som er lavere enn terskelverdien hører til bakgrunn (0), og de som er høyere hører til forgrunn (1) eller vica versa. Nedenfor er kriterier for en slik terskling gjengitt, der $g(x, y)$ er det tersklede bildet, og $f(x, y)$ er originalbildet. Resultatet for hvordan dette kan bli seende ut er vist i figur 2.3

$$g(x,y) = \begin{cases} 1 & \text{hvis } f(x,y) > \mathrm{T} \\ 0 & \text{hvis } f(x,y) \leqslant \mathrm{T} \end{cases}$$

Global terskling kan også gjøres på fargebilder. Ved å gjøre om RGB-bilde til gråtonebilde kan tersklingen utføres. Dette fører til dårlige resultater [4]. Derimot kan terskling gjøres på de tre verdiene som beskriver hvert piksel. Det må da settes tre grenser for å sette en global terskel for fargebilder. Disse grensene kan være forskjellige fordi det er tre separate verdier som ikke er avhengige av hverandre. Dette gjør at man kan sette klare grenser for hvilke farger som skal karakteriseres som forgrunn. Ved å terskle et bilde på denne måten kan man få gode resultater [4, 13, 21].



(a) Gråtone (b) Tersklet bilde

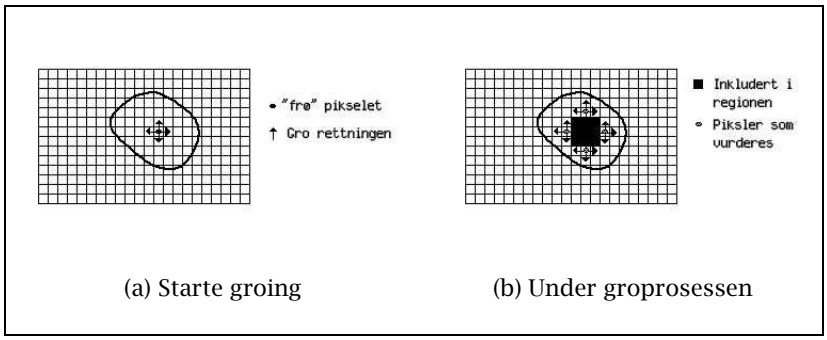
Figur 2.3: Illustrasjon av terskling [33]

2.2.2 Regionbasert segmentering

Regionbasert segmentering finner hele regioner. Regionene kan ha forskjellige krav ut fra hva som er ønskelig å finne i et bilde. Denne formen for segmentering finner hele regioner som tilfredsstiller kravet som er satt.

Groing av regioner

Prinsippet for groing av regioner går ut på at alle piksler er en egen region. Krav til regioner kan være gråtoner, farger, tekstur eller form. Piksler som tilfredsstiller krav og er plassert ved siden av hverandre, smeltes sammen og blir en region. For å finne slike regioner, blir det plassert ut "frø" i bildet. Plasseringen av disse "frøene" kan enten gjøres vilkårlig eller ved at man ønsker en region som et bestemt piksel tilhører. Ut fra "frøet" blir piksler som tilfredsstiller kravet til regionen inkludert. Regionen vokser helt til det ikke er flere piksler som grenser til regionen som tilfredsstiller kravet. En generell groing er vist i figur 2.4.



(a) Starte groing (b) Under groprosessen

Figur 2.4: Illustrasjon av groing [15]

Region splitting

Region splitting er det motsatte av region groing. Her ansees i utgangspunktet hele bildet for å være en region. Det er sjelden hele bildet tilfredsstiller kravet man har til en region. Regionen splittes i to og de nye regionene blir undersøkt. Dette foregår til man har regioner man ønsker. Det er sjelden man kommer frem til de ønskede regionene bare ved å splitte. Derfor kombinerer man splitting ofte med sammenslåing av regioner. Dette fører til at regioner som er plassert ved siden av hverandre kan settes sammen dersom de er tilstrekkelig like.

2.2.3 Kantbasert segmentering

I motsetning til regionbasert segmentering, finner kantbasert segmentering bare overgangen mellom regioner. Kantbasert segmentering er en fellesbetegnelse for en stor gruppe metoder som segmenterer bilder ut fra kantene i bildet. Disse metodene er stort sett laget for gråtonebilder. Ved å gjøre om fargebilde til gråtone bilde kan enkelte detaljer forsvinne. Metodene baserer seg på informasjon om et begrenset område i bildet, og ut fra dette finner den grensen mellom forskjellige objekter.

Ut fra regioner er det lett å finne kanter, og ut fra kanter er det lett å finne regioner [23]. Jeg har ikke funnet noen kantbaserte segmenteringsmetoder som finner de regionene gjenkjenningssystemet er ute etter uten at regions segmentering også gjør det. Siden systemet som skal utvikles er basert på å finne regioner, er det ikke valgt å se nærmere på kantbasert segmentering.

2.2.4 Valg av segmenteringsmetode

Når metode for segmentering skal velges, er det avgjørende å benytte den informasjonen man har om bildet og hvilke elementer i bildet man er opptatt av å identifisere [9, 23]. Som nevnt skal det utvikles et system som skal gjenkjenne *norske* fartsskilt ut fra stillbilder. Vi vet at disse skiltene har svarte siffer på hvit flate med rød sirkel rundt.

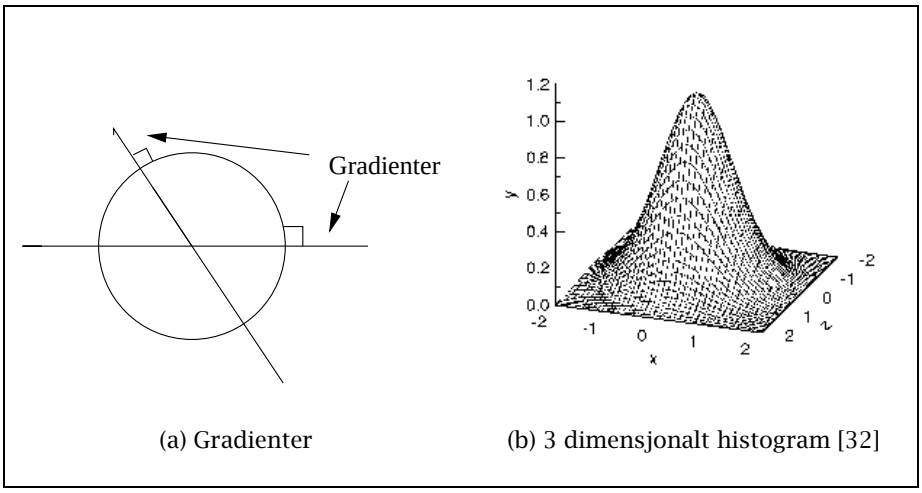
Det letteste å gjenkjenne i et norsk fartsskilt er den røde sirkelen. Den har farge som ikke mange andre objekter i naturen har. Informasjon om fargen som ligger inntil denne røde sirkelen er også kjent. For at all denne informasjonen skal benyttes, er det valgt å gjøre segmentering ved hjelp av groing av regioner. Dette er en meget rask metode som det er mulig å begrense på en effektiv måte. Omgivelsen til sirkelen benyttes til å redusere objekter som kan tilhøre fartsskilt. Det er valgt å legge "frø" bare i de røde pikslene som har hvitt nabopiksel. Dette forklares nærmere i avsnitt 3.3.2.

2.3 Lokalisere sirkel

Det finnes flere metoder for å undersøke om det eksisterer sirkler i et bilde. Jeg vil her beskrive enkelte metoder som kan benyttes for å lokalisere skilt.

2.3.1 Hough-transformasjon

Det er flere måter Hough-transformasjonen kan benyttes for å finne sirkler. Her vil bare en av disse bli beskrevet [17]. Gradienten til alle kantene i et filtrert bilde blir regnet ut og registrert i et 3-dimensjonalt histogram. En gradient er en linje som krysser en linje med en vinkel på 90°. Dette er illustrert i figur 2.5 (a). Blir det funnet punkt i histogrammet der mange gradienter passerer, anses dette for å være sentrum i skiltet, dette er vist i figur 2.5 (b). Ved å benytte denne metoden, er det mulig å lokalisere skilt som er delvis dekket av andre objekter.



(a) Gradienter (b) 3 dimensjonalt histogram [32]

Figur 2.5: Illustrasjon av Hough-transform

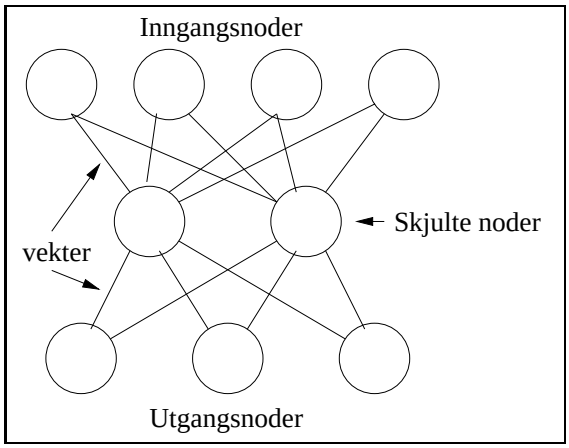
2.3.2 Nevrale nett

Kunstig nevralt nett er en etterlikning av måten menneskers hjerne er bygget opp på. Nettet består av inngangs- og utgangsnoder som er koblet sammen gjennom skjulte noder. Koblingene mellom nodene er vektet. Vektingen beskriver hvor mye en kobling skal vektlegges i forhold til de andre koblingene. Disse vektene "læres opp" ved at systemet trenes med mange input hvor utfallet alt er kjent. Koblingene blir gitt vekt slik at riktig utgangsnode blir prioritert. Eksempel på slikt system er vist i figur 2.6. Nevrale nett er brukt som metode for å finne sirkler i [19].

2.3.3 Templat

For å finne et bestemt ønsket mønster eller objekt, kan templater benyttes. Et templat er en beskrivelse av objektet man ønsker å finne, og kan sees på som en mal. Dette templatet blir så ført over hele bildet for å finne ut om den passer et sted i bildet. Dersom den passer tilstrekkelig, vil det bli konkludert med at objektet er funnet.

Ved å justere kravet til hvor stor del av templatet som skal passe, kan også denne metoden lokalisere objekter som er delvis skjult. Det er også mulig å bestemme størrelsen på objektene som skal lokaliseres ved å benytte templater med forskjellig størrelse. For skiltgjenkjenning vil innholdet i skilt være umulig å tyde dersom skiltet er under en viss størrelse. Templater egner seg godt til å overholde en slik grense. Dette er den



Figur 2.6: Et nevralt nett

metoden som oftest benyttes i systemer som er laget for å gjenkjenne skilt [18, 21, 28].

2.3.4 Begrunnelse for valg av metode for å finne sirkler

Det kan gi gode resultater å lokalisere skilt ved å benytte Hough-transformasjon, men den har også noen svakheter. Dersom det er røde objekter bak skiltet, kan disse bli oppfattet som en del av selve skiltet. Dersom det er montert skilt med rødmalt hus i bakgrunnen, kan huset oppfattes som del av selve skiltet. Dette virker inn på gradienthistogrammet, og kan gjøre at skilt ikke blir funnet. Hastigheten på metoden vil være avhengig av at filtreringen tar bort det aller meste som ikke er fartsskilt. Dersom dette ikke er tilfredsstilt, vil det føre til mange regneoperasjoner for å finne alle gradientene. Dette kan gå utover hastigheten for gjenkjenningen. Metoden finner sentrum av skiltet, men størrelsen kan ikke fastslås.

Dersom nevralt nett skal benyttes, er det ikke mange mulige måter å gjøre dette på. Potensielle skilt normaliseres til 32x32 piksler og brukes som input til nettet [19]. I og med at alle inngangsnoder er koblet til alle de skjulte nodene, blir det mange regneoperasjoner når det er 1024 innganger. Dersom det er mange objekter som skal sjekkes, vil dette føre til stort tidsforbruk for å lokalisere sirkel.

Høyst sannsynlig skal lokalisering av skilt implementeres på FPGA. Nevrale nett egner seg ikke godt til dette, fordi alle koblingene mellom

nodene krever meget nøyaktige verdier for å komme frem til riktig resultat. Dette krever stor plass i en FPGA.

Siden det bare skal lokaliseres én geometrisk form, er bruk av templat en meget effektivt metode [13, 21]. Templatmetoden kan også meget effektiv implementeres på FPGA [21]. Dette er begrunnelsen til at templater er valgt som metode for å lokalisere fartsskilt i IDSL.

2.4 Tyde siffer

For å fastsette hvilken fartsgrense som er angitt på skilt er det nødvendig å tyde sifferet. I de følgene avsnittene vil jeg gi en innføring i forskjellige slike metoder.

2.4.1 Templat

Templat kan benyttes til å gjenkjenne objekter, som beskrevet i avsnitt 2.3.3. Malen av det som skal lokaliseres føres over bildet. Blir det funnet objekter med tilstrekkelig likhet, blir dette ansett for å være objektet. For skiltgjenkjenningssystemer er denne metoden benyttet i [17]. Der benyttes templater som beskriver hele skiltet, altså ikke bare sifferet.

2.4.2 Egenskaper

Det er mulig å gjenkjenne forskjellige objekter ved å teste enkelte egenskaper ved objektene. Enkelte objekter kan se ut som tall uten å være det, mens tall kan forveksles med andre objekter. Det vanskelige med denne metoden, er å finne ut hva som sikkert kan skille tall fra andre objekter. I [21] er denne metoden benyttet for å gjenkjenne siffer.

2.4.3 Nevrale nett

Nevrale nett (NN) er beskrevet i avsnitt 2.3.2, og identifiseringen går ut på at man har objekter man sender inn i nettet. Nettet er “lært opp” ved hjelp av trening til å gjenkjenne enkelte mønstre, og gir beskjed om hvilket av disse mønstrene objektet som testes likner mest på.

2.4.4 Evolusjonær maskinvare

Evolusjonær maskinvare (EHW) er beskrevet i [27]. Den største forskjellen mellom EHW og nevrale nett, er at EHW "læres opp" ved hjelp av evolusjon.

2.4.5 Begrunnelse for valg av metode for å tyde siffer

Templater er benyttet for å gjenkjenne siffer i skiltgjenkjenningsmetoden som er presentert i [17]. Resultatet av denne metoden er ikke tilstrekkelig. Under 50% av sifrene blir gjenkjent. Det er derfor valgt å benytte en annen metode for å gjenkjenne siffer i denne oppgaven.

I [21], som denne oppgaven bygger på, benyttes egenskaper ved sifrene for å identifisere dem. Sifferet normaliseres, og for at et siffer skal bli tydet, må enkelte deler av sifferet stemme med matriser som er laget for å beskrive forskjellige tall. Det er gjort forsøk på å gjenkjenne siffer ved å benytte denne metoden.

NN og EHW har gode resultater for gjenkjenning av siffer [26, 27]. Det er derfor valgt å se nærmere på disse metodene for å gjenkjenne siffer.

Kapittel 3

Image Detection of Speed Limits

Dette kapittelet beskriver bearbeidingen av bilder for å fastslå om de inneholder fartsskilt, og eventuelt hvilken fartsgrense skiltet informerer om. Hele denne beskrivelsen er basert på den opprinnelige versjonen av IDSL, som er utviklet av Sekanina og Tørresen [21]. Denne versjonen har klare svakheter både når det gjelder tidsforbruk og gjenkjenning. Som nevnt i kapittel 1, har jeg laget en tidtakermetode som tar tiden på de forskjellige fasene av gjenkjenningsprosessen. Enkelte steder i dette kapittelet kommenteres tidsforbruket og disse resultatene er hentet fra denne metoden, som forklares nærmere i avsnitt 4.2.2. Mine forslag til forbedringer er beskrevet i kapittel 4.

Fartsskiltene i Norge varierer i spennet fra 30 km/t opp til 90 km/t med et intervall på 10 km/t. Det er for tiden under utprøving med fartsgrense på 100 km/t for enkelte strekninger, men det er ikke vedtatt endelig godkjenning for dette skiltet enda. En annen type fartsskilt er opphevingsskilt. Disse er det heller ikke laget gjenkjenning for enda. Oversikt over norske fartsskilt er vist i figur 3.1.

3.1 IDSL sin oppbygning

IDSL prosessen kan grovt sett deles inn i fire faser. Første fase er å lese inn bildet. Den neste fasen er fargefiltrering som identifiserer alle de røde områdene i bildet. Fase tre har som formål å identifisere om det finnes et fartsskilt ved å søke etter sirkler i det filtrerte bildet. Dersom det blir funnet sirkel, er siste fase å analysere innholdet i sirkelen for å fastslå hvilken fartsgrense skiltet informerer om.

Figur 3.1: Oversikt over norske fartsskilt [26]

Figur 3.2: Brukergrensenittet til IDSL

Grensesnittet til IDSL er vist i figur 3.2, med bildet som skal prosesseres. Under bildet er det informasjon om plasseringen til pikselet musepekeren befinner seg over, og hva RGB og HSI-verdiene er for dette pikselet. Under dette står kriteriene for at et piksel skal karakteriseres som rødt, hvitt eller svart. Den store knappen (Find Speed Limit) starter selve gjenkjenningsprosessen, som sjekker bildet som vises. Under den store knappen vises resultatet av gjenkjenningen når den er gjort. Det åpne feltet helt nederst er beregnet på informasjon som IDSL gir om gjenkjenningsprosessen. De to siste knappene gir mulighet til å vise filtrerte bilder som IDSL lager og bruker under gjenkjenningen (figur 3.4 og figur 3.7).

3.2 Innlasting av bildet

Et bilde som skal analyseres, må først lastes inn fra fil før gjenkjenningen kan begynne. Under innlastingen blir RGB-verdiene skilt fra hverandre, og plassert i tre separate matriser. Matrisene er like store som originalbildet (640x480) og inneholder verdiene til hver av fargene rødt, grønt og blått for hvert piksel. Originalbildet blir ikke brukt mer, så all informasjon hentes fra disse matrisene. Under innlesningen blir gjennomsnittsverdien til pikslene i bildet regnet ut. Denne verdien er et estimat for hvordan lysforholdet var da bilde ble tatt. For at fartsskilt skal virke så like som mulig for IDSL, er det en fordel at lysforholdene er like. Dette er ikke oppnåelig når bildene tas utendørs. Gjennomsnittet brukes til å avgjøre om et bilde er for *mørkt*, og om det dermed skal gjøres justeringer av bildet for å få det lysere. Det er satt en grense slik at gjennomsnittlig pikselverdi ikke skal være lavere enn 125. Dersom verdien er under denne, blir verdiene i alle pikslene i bildet justert like mye som differansen mellom gjennomsnittet og grenseverdien. Dette betyr at dersom gjennomsnittlig pikselverdi er 123, blir verdien til alle pikslene i hele bildet økt med 2 slik at gjennomsnittet blir 125.

Det er tre ulemper med slik innlastingen gjøres. En ulempe er at det ikke er mulig å gjenkjenne mer enn ett bilde av gangen. Bildet må velges manuelt for hver gang det skal testes. Den andre ulempen er at metoden som henter RGB-verdiene til pikslene gjør dette ut fra bildet som vises på skjermen. Det er en tidkrevende metode å skaffe denne informasjon på. Den siste ulempen er at det bare er laget grenser for hvor mørke bilder kan være, men ikke for hvor lyse bildene kan være før de må justeres.

3.3 Fargefiltrering

Måten IDSL finner skilt på er, som tidligere nevnt, å lete etter den røde sirkelen som omkranser skiltet. Dette betyr at de røde pikslene i bildet må lokaliseres. Det kan være veldig mange røde piksler i bildet, men alle hører ikke til fartsskilt. Derfor må mengden områder som potensielt kan være fartsskilt begrenses. Dette gjøres ved å benytte informasjon om skiltets utseende. Norske fartsskilt har en rød ring rundt en hvit flate med svarte siffer[1] på. Områder som blir tatt med videre i analyseringen begrenses til de røde områdene som har *hvitt* som nabo. Det er også beskrevet andre gjenkjenningssystemer i litteraturen som benytter seg av et slikt kriterium [17].

3.3.1 Fargenes grenseverdier

IDSL benytter fargene rød, hvit og svart for å gjenkjenne et fartsskilt. For at det skal være mulig for systemet å gjenkjenne skilt, må disse fargene defineres. Kravet for at et piksel skal karakteriseres som hvitt, er at alle tre RGB-verdiene hver for seg må være større enn 150. For at et piksel skal karakteriseres som svart, må alle RGB-verdiene være mindre enn 180. Kriteriet for at et piksel er rødt, er at R-verdien må være større enn 80 og minst 20 større enn både G og B-verdiene:

- Pikselet er hvitt hvis: R > 150 og G > 150 og B > 150
- Pikselet er svart hvis: R < 180 og G < 180 og B < 180
- Pikselet er rødt hvis: R > 80 og G < R-20 og B < R-20

Som grensene viser, kan enkelte piksler både betrakte som svarte og hvite. Dette går ikke utover gjenkjenningen av fartsskilt fordi det er forskjellige faser som benytter disse to fargene. Enkelte piksler kan bli sett på som både hvitt og rødt. Dette kan føre til at enkelte røde piksler blir sett på som hvite fremfor røde. Dette kan føre til at deler av den røde ringen karakteriseres som hvit fremfor rød.

Grensene for de forskjellige fargene er laget på bakgrunn av et utvalg av bilder som er tatt under en og samme type værforhold. Dette fører til at grensene virker godt på bilder som er tatt i samme lysforhold. Derimot

[1] Gjennom oppgaven benevnes tall på skiltet som siffer

er grensene lite egnet for bilder som er tatt under andre lysforhold. Det viser seg at enkelte gulfarger tilfredsstiller kravet til rødt. Dette fører til at templater testes på flere punkter enn nødvendig. Det er også større sjanse for at objekter som ikke er fartsskilt sendes videre i gjenkjenningsprosessen. Dette legger beslag på systemets ressurser og forsinker prosessen unødvendig.

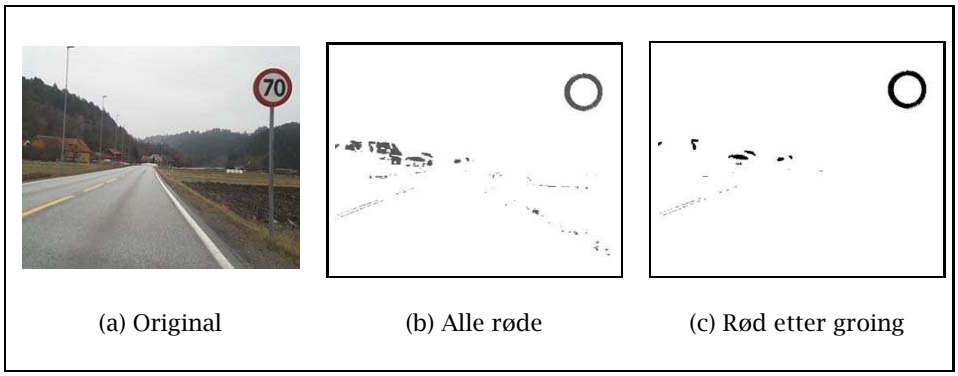
3.3.2 Segmentering

Som nevnt i avsnitt 2.2.4, benyttes groing med kontrollert plassering av "frø" for å segmentere bildet. I IDSL gjøres dette ved først å lokalisere 2x2 matriser med *hvite* piksler. Hver gang en slik matrise blir funnet, sjekkes det om nabopikslene til matrisen er røde. Dersom noen av nabopikslene er røde og ikke alt markert, plasseres "frøet" i det røde pikselet. Resten av regionen til det røde pikselet blir funnet ved å "gro" regionen. Teorien bak "groing" av regioner er forklart i avsnitt 2.2.2. Når dette er gjort forsetter søket etter 2x2 matriser med hvite til hele bildet er sjekket. Også andre skiltgjenkjenningssystemer benytter seg av denne metoden [28].

Regionen blir "grodd" ved å benytte rekursivt kall. Dette er en effektivt algoritme som ofte benyttes under digital bildebehandling. Rekursivitet kan kort forklares ved at det er en metode som kaller seg selv. Rekursjon brukes på problemer som er komplekse, men som kan deles opp i mindre deloppgaver som lettere kan løses. For IDSL vil det si at oppgaven er å finne alle pikslene som tilhører samme regionen som "frø"-pikselet. Dette gjøres ved at "frø"-pikselet sjekker alle naboene, som i vanlig rekursjon, og avgjør om de tilfredsstiller kravene for å være rødt. Dersom noen gjør det, sjekkes alle naboene til dette pikselet igjen. Slik går det helt til hele regionen er funnet, og ingen piksler i regionen har nabopiksler som tilfredsstiller kriteriene for rødt.

Kriteriet for at et piksel skal innlemmes i regionen er at det er rødt. Pikslene som blir funnet under groingen registreres i en ny matrise (rødmatrise) som er like stor som originalbildet. Når metoden er ferdig gjennomført, sitter man igjen med rødmatrisen som inneholder de røde piksler som kan tilhøre sirkelen på et fartsskilt. Det eneste som er funnet så langt i prosessen er de røde regionene, som har en eller flere hvite nabopiksler. Formen på regionene er ikke fastlagt enda.

I figur 3.3 vises tre bilder. Det første (a) er originalbildet. Bilde (b) viser alle pikslene i bildet som tilfredsstiller kravene for rødt. Det siste bildet



(a) Original (b) Alle røde (c) Rød etter groing

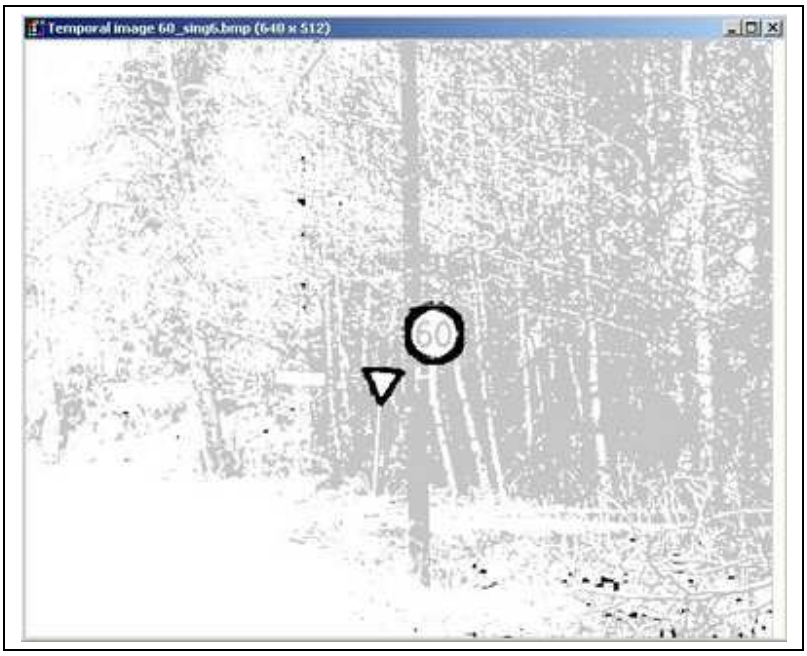
Figur 3.3: Illustrasjon på resultat etter groing

(c) viser redusert matrise. Den inneholder de røde pikslene som er markert etter å ha grodd regioner. De hvite delene av bildet markeres også i denne matrisen, men disse blir ikke benyttet videre. En filtrert versjon av bildet i figur 3.2 er vist i figur 3.4. De røde pikslene er markert som sort og de hvite som hvitt, mens det grå verken er rødt eller hvitt.

Med den rekursive metoden sjekkes alle naboene til alle pikslene selv om det er klart at en av naboene alt er testet. Hver sjekk bruker ikke lang tid, men den gjøres så mange ganger at en optimalisering av algoritmen kan spare betraktelig med tid.

Oppsummering av fargefiltreringen:

1. Søk etter 2x2 matrise med hvite piksler. Blir hvite funnet, gå videre til 2. Hvis ikke, fortsett søk.
2. Det sjekkes om nabopikslene til 2x2 matrisen er røde. Dersom en av dem er røde gå videre til 3. Hvis ikke, gå tilbake til 1.
3. Dersom pikselet ikke alt er markert i rødmatrisen, plasseres “frø” i pikselet. Ved å “gro” blir resten av regionen funnet, som deretter markeres i rødmatrisen.
4. Punkt 1 gjentas til hele bildet er sjekket.



Figur 3.4: Filtrert bilde

3.4 Lokalisere skilt

Formålet med denne fasen av IDSL er å finne posisjonen og størrelsen til potensielle fartsskilt i bildet. Som nevnt i forrige kapittel (avsnitt 2.3.4), er det valgt å benytte templater til denne lokaliseringen.

3.4.1 Templatene

Et templat er en mal, i dette tilfelle en sirkel som benyttes for å vurdere om det finnes formlike objekter i bildet. Det er forskjellig størrelse på templatene fordi størrelsen på skiltene er avhengig av avstanden mellom kameralinsen og skiltet. Siden det skal være mulig å lokalisere skilt på flere avstander, må det benyttes templater med forskjellig størrelse. Templatene som benyttes for å lokalisere skilt, varierer i størrelse fra 20x20 piksler til 160x160 piksler. Det benyttes 10 templater som ligger i dette intervallet. Matrisen som templatene testes på er rødmatrisen. I avsnitt 3.3 er det forklart hvordan rødmatrisen lages. Alle templatene blir ført over rødmatrisen, og det blir sjekket hvor stor andel av templatene som passer på objektene i rødmatrisen. Hver gang en templat og

objektet er 50% like eller mer, gjøres tester for å avgjøre om det kan være fartsskilt. Disse testene er forklart i avsnitt 3.4.2. Ved å teste templater på rødmatrisen begrenses antall teststeder. For å begrense antallet ytterligere, blir templatene bare testet på hvert *annet* piksel. Dette halverer antall punkter. Dette har ingen betydning for kvaliteten eller gjenkjenningsresultatene. Det er foretatt tester som bruker alle pikslene. Disse testene viste at resultatet er det samme som når bare hvert annet piksel testes. Alle templatene blir testet og prosentandelen av templatene som passer blir registrert.

Det viser seg at enkelte templater aldri passer på skilt det er tatt bilde av. Dette vil si at det går med mye tid på å sjekke alternativer som normalt aldri vil forekomme. Dette beskrives nærmere i avsnitt 4.5.1

3.4.2 Testing av templatets plassering

Denne fasen av IDSL undersøker om det virkelig er et fartsskilt når templat har gitt treff. Det er mange metoder som er flettet sammen. Testene som gjøres skal sikre at objekter som likner på fartsskilt, men ikke er det, blir eliminert på et tidlig stadium i gjenkjenningen. Det testes om området templatet ligger over har tydelige egenskaper som entydig sammenfaller med egenskaper til fartsskilt. Siden testene gjøres så mange ganger, er det viktig at testene som gjøres er sikre og raske.

For hver gang en templat stemmer tilstrekkelig, blir det foretatt tre enkle tester for å finne ut om det kan være et fartsskilt. Det blir sjekket om det finnes røde piksler rundt sentrum av skiltet. Dersom ett slikt piksel blir funnet, forkastes forslaget og søket etter steder der templatet passer fortsetter. Testen forhindrer at forbudsskilt som parkering forbudt, forbud mot motorvogn og andre skilt med rød strek på tvers av skiltet blir oppfattet som fartsskilt og blir behandlet videre. For å forhindre at trekantet skilt blir sett på som røde, gjøres test som forsikrer om at det er omtrent like mange røde piksler i alle fjerdedelene av templatet. Den siste testen sjekker at andelen av hvitt i området templatet dekker ligger mellom 10% og 95%. Dette gjøres for å utelukke at templatet feiltolker en homogen flate til å være skilt. Testene som gjøres med området for å sjekke om området templatet ligger over er et fartsskilt kan oppsummeres slik:

- Test sjekker at det ikke finnes røde piksler i midten av templatet. Dette blir gjort for å ekskludere forbudsskilt. Eksempel på dette er vist i figur 4.2 (d).

- Test sjekker om det er like mye rødt i alle fjerdedelene av templatet. Dette gjøres for å ekskludere trekantede skilt.
- Test sjekker at andelen hvitt i hele skiltet ligger mellom 10% og 95%.

Dersom en av testene feiler, blir det umiddelbart forkastet som fartsskilt, og søket fortsetter. Etter at alle templatene er sjekket på annethvert piksel, betraktes området hvor templatet som passer best som det interessante område. Dette området analyseres videre. Blir det ikke funnet templat som passer tilstrekkelig, konkluderes det med at det ikke finnes fartsskilt i bildet.

Testene som gjøres hver gang et templat testes på en posisjon er ikke tilstrekkelig til å forkaste alt som ikke er fartsskilt. Skilt som forbud mot alle kjøretøyer og innkjøring forbudt er skilt som ikke forkastes av testene over. Feltet hvor det sjekkes at det ikke er rødt i templatet er større enn nødvendig for å være sikker på at det ikke er forbudsskilt. Tidsforbruket for denne fasen av IDSL er veldig varierende. Tidsmålinger viser at de største templatene bruker lenger tid på å bli testet enn de små.

Oppsummering av lokalisering av skilt:

1. Hvert annet piksel i rødmatrisen testes med alle templatene.
2. For hver gang en templat passer bedre enn 50%, vil nye tester avgjøre om det kan være fartsskilt.
3. Området hvor templatet passet best, betraktes som interessant og analyseres videre. Passer ingen templater, konkluderes det med at det ikke finnes skilt i bildet.

3.5 Lokalisere og tyde sifferet

Etter at templatet som passet best er funnet, gjenstår det å finne ut hvilke siffer som står på skiltet. For norske fartsskilt vil alltid det siste sifferet være null, så tydingen er fokusert på det første sifferet. Denne fasen av IDSL opererer på et meget begrenset området av bildet, nemlig *inne* i det området hvor templatet passet best.

3.5.1 Klargjøre området

Det første som gjøres er å kvalitetssikre det interessante området som er funnet i avsnitt 3.4. I dette området gjøres hvite piksler som er inneklemt mellom to røde, om til rødt. Disse "hullene" i ringen kan være forårsaket av blant annet støy. Dette gjøres for å klargjøre det interessante området slik at sifrene kan trekkes ut fra området. Det er helt avgjørende at områdene i templatet er homogene for at sifrene skal trekkes ut på en sikker måte. Neste skritt er å fjerne alt annet enn sifrene i skiltet fra området.

Det er ikke bare hvite piksler som kan forårsake "hull" i det røde området. For enkelte skilt kan det forekomme at ikke hele den røde ringen betraktes som rød, men feilaktig betraktes som svart. Grunnen til dette kan være lysforholdene eller at selve skiltet er skittent. Dette kan føre til feiltolkning av sifferet.

3.5.2 Trekke ut sifrene

Det stilles strenge krav for at et piksel skal bli definert som en del av siffer. På hver linje i templatet må det bli funnet rødt piksel før piksler kan tilhøre siffer. Svarte piksler som så blir funnet, ansees for å være en del av et siffer så lenge det kommer rødt piksel etter det svarte på den samme raden.

Krav for at svart piksel skal tilhøre siffer på en linje kan oppsummeres slik:

1. Det må finnes rødt piksel på en linje før punkt 2 utføres.
2. Svarte piksler markeres som siffer.
3. Dersom det ikke blir funnet rødt piksel før kanten av templatet, blir hele linjen blanket ut.

Etter at hele templatet er sjekket på denne måten er det markert en rekke svarte piksler. Disse svarte pikslene ekstraheres, og benyttes videre for å komme frem til fartsgrensen.

Enkelte templater kan ligge så forskjøvet på skiltet at ikke hele den røde sirkelen ligger på innsiden av templatet. Dette gjør det mer krevende for systemet å trekke ut sifrene fra skiltet. I figur 3.5 er et eksempel på

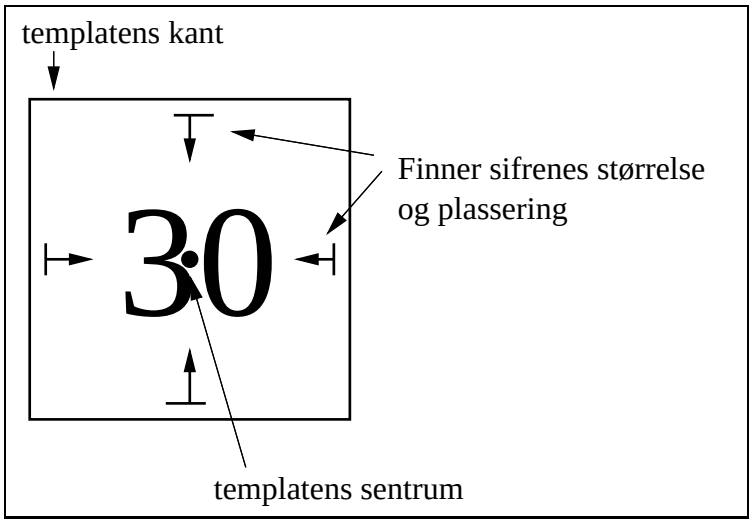
Figur 3.5: Deler av nummeret blir blanket ut

hvordan dette påvirker uttrekkingen av sifferet. Her deles sifrene i to fordi det ikke er rødt piksler på linjen til venstre for sifrene.

3.5.3 Størrelse og plassering av sifrene

Størrelsen og plasseringen til sifrene skal så finnes. Et søk går gjennom hver pikselrad fra templatets ytterkant mot sentrum. Det blir undersøkt om det finnes svarte piksler i disse radene. Dersom det finnes tre svarte piksler på en rad, anses dette for å være starten på sifrene. Grunnen til at det må være over tre på en linje før det anses å tilhøre et siffer, er for å unngå at små svarte objekter på skiltet anses for å være starten av siffer. Ved å gjenta søket fra alle kanter vil størrelsen og plasseringen av sifrene være funnet. Dette er illustrert i figur 3.6. De to sifrene deles fra hverandre, ved å dele området midt mellom de to vertikale ytterpunktene. Som nevnt er det høyre sifferet alltid null på alle norske fartsskilt. Dette blir ikke tatt med videre i analysen. Størrelsen og plasseringen til det første sifferet er funnet, og det gjenstår bare å identifisere hvilket tall det er.

Selv om det er tatt høyde for at enkelte små objekter som ligger på utsiden av sifrene skal ignoreres, skjer ikke dette alltid. Dette fører til at området IDSL mener sifrene befinner seg i, er større enn det virkelig er. Dette kan føre til at skillet mellom sifrene går gjennom ett av sifrene. Det er ingen begrensning på størrelsen et siffer kan ha. Det kan føre til at enkelte templater som ligger over områder i bildet som ikke er fartsskilt, kan ansees for å være fartsskilt. Objekter som ikke er siffer blir dermed forsøkt tydet.



Figur 3.6: Finne størrelse og plassering av sifrene

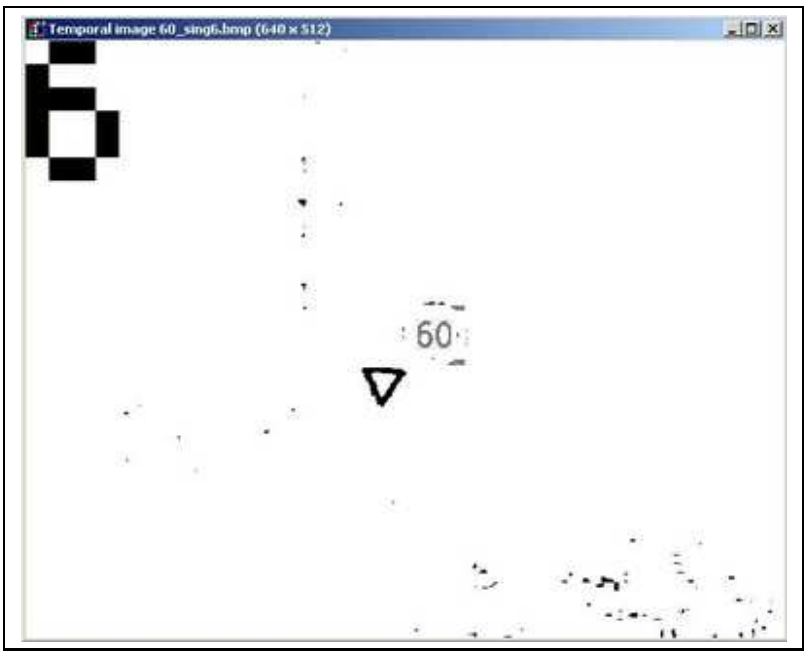
3.5.4 Tyde sifferet

Tyding av sifferet gjøres ved å normalisere sifferet til en 7(rader) x 5(kolonner) matrise. Denne normaliseringen gjøres ved at områdene får farge ut fra om det er mest svart eller mest hvitt i det respektive område. Denne normaliseringen gjør det lettere å gjenkjenne sifferet. Et slik siffer kan sees i øverste venstre hjørne av figur 3.7, som er resultatet av filtreringen av bildet som vises i figur 3.2.

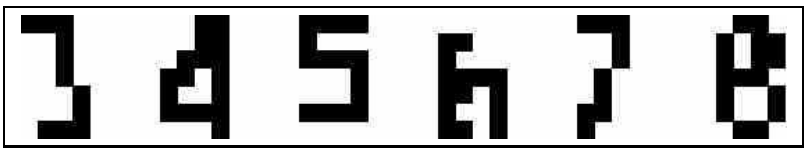
For å finne ut hvilket tall det første sifferet er, blir matrisen sammenliknet med en matrise som representerer egenskaper for forskjellige tall. I den originale versjonen av IDSL er det bare laget slike matriser for 5 og 6. Dersom en av disse matrisene likner tilstrekkelig mye på kandidatmatrisen [21], vil det konkluderes med at sifferet er identifisert. Når sifferet er identifisert, konverterer systemet tallet til fartsgrense. Dersom det ikke blir funnet egenskapsmatriser som stemmer tilstrekkelig, vil konklusjonen være at IDSL ikke er i stand til å tyde fartsgrensen. Enkelte siffer som skal sammenliknes med egenskapsmatriser er vist i figur 3.8

Sammenlikning av det normaliserte sifferet med egenskaper til tall er enkelt for maskinen. Det er derimot vanskelig å lage gode slike matriser. Det er bare laget matriser for å tyde tallene 5 og 6. Det er dermed mange fartsskilt som det er umulig å tyde.

Oppsummering av lokalisering og tyding av sifrene:



Figur 3.7: Ekstrahert siffer



Figur 3.8: Siffer fra gjenkjenning [26]

1. “Hull” blir tettet, for å få homogene områder. Alt annet enn det svarte inne i skiltet blir fjernet.
2. Bredde og høyde på sifrene blir funnet, det siste sifferet (0) blir fjernet.
3. Det første sifferet blir normalisert til en 7x5 matrise, som blir sammenliknet med egenskaper til forskjellige tall.
4. Har den normaliserte matrisen tilstrekkelig like egenskaper med et tall, indikerer dette fartsgrense. Ellers vil det konkluderes med at IDSL ikke er i stand til å tyde fartsgrensen.

Kapittel 4

Forbedret IDSL

Dette kapittelet inneholder forandringer jeg har gjort med metodene som er forklart i kapittel 3 og diskusjon rundt disse forandringene. Det er flere grunner til at forandringene er gjort:

- Forbedre kvaliteten på gjenkjenningen.
- Reduksjon av tidsforbruk. For at IDSL skal være mest mulig pålitelig er det viktig at flest mulig bilder blir analysert. Gjenkjenningen bør derfor gå så raskt som mulig, uten at det går ut over evnen til å gjenkjenne skilt.
- Kontinuitet og forutsigbarhet. Uansett hvordan bildet ser ut, bør tiden som brukes på hvert bilde være så lik som mulig.
- Tilnærming til endelig løsning. Foreløpig er IDSL en simulering, men i fremtiden kommer den til å bli koblet til et kamera. Det er derfor viktig at simuleringen oppfører seg så likt et kamera som mulig. IDSL skal oppføre seg på samme måte når et kamera er koblet til.
- Lokalisere hvilke deler som er mest tidkrevende, for å finne ut hva det er aktuelt å implementere i maskinvare.

4.1 Bildene

For at simuleringen skal være så realistisk som mulig, er det viktig at stillbildene som benyttes er så naturtro som mulig og så nær den situa-

sjonen og det miljøet som IDSL vil bli brukt i. Derfor er alle bildene jeg tatt fra passasjersetet i en bil. Det vil være på passasjersiden av bilen kameraet skal plasseres. For at testen skal være så realistisk som mulig er det ønskelig med store mengder bilder, men dette er ikke avgjørende. Det viktigste er at bildene dekker de situasjonene som kan inntreffe når IDSL skal tas i bruk. Kameraet som er brukt til å ta bilder er Nikon Coolpix 995. I og med at det ikke er klart hvilken type kamera som kommer til å bli plassert i kjøretøyet, er heller ikke kvaliteten og størrelsen på bildene som skal brukes fastlagt. Det er valgt å benytte den minste oppløsningen som er mulig med kameraet. Alle bildene har oppløsning på 640x480 piksler. Bildene er i komprimert jpg format, og gjøres om til bmp, siden det er dette formatet IDSL tar som input.

4.1.1 Været og tidspunkt

Som nevnt i kapittel 1, har været stor betydning for gjenkjenningen. Det er været som avgjør hvor mye lys det er i omgivelsene. Dette har igjen direkte innvirkning på fargebeskrivelsen av omgivelsene. Dersom det er lite lys, vil mindre lys bli reflektert fra objektet. Dette gjør at fargene oppfattes mørkere. Hvis det er mye lys, vil fargene oppfattes lysere. For å få en realistisk test er det viktig at de bildene som brukes er tatt under forskjellig værforhold. Det er tatt bilder under tre forskjellige værtyper: sol, lett skydekke og tungt skydekke med noe regn. Ved å bruke bilder tatt under disse tre forholdene, vil de vanligste forekommende lysforholdene bli gjengitt. Det er ikke foretatt tester for å gjenkjenne skilt om natten. Grunnen til dette er at det nattestid vil langt svakere lys enn på dagtid. Dette vil vanskeliggjøre fotograferingen og dermed gjenkjenningen. Det vil her være nødvendig å vurdere tilgjengelig kamerateknologi for å fastslå om gjenkjenning i mørket er mulig. Kompleksiteten i trafikkbildet er også helt annerledes om natten. Behovet for IDSL er størst om dagen. Dette er grunnen til at jeg har konsentrert meg om gjenkjenning i dagslys.

4.2 Testing

For å teste IDSL sin evne til å gjenkjenne fartsskilt, brukes det to tester. Dette gjøres for å se hvordan forandringer i koden påvirker gjenkjenningen. Den ene testen finner ut hvordan forandringene påvirker evnen til å gjenkjenne fartsskilt. Dette betyr at etter forandringer er gjort, sammenliknes gjenkjenningsresultatene med hvordan de var før forandrin-

gen. Det andre som vurderes er *tidsforbruket*. Det blir sett på hvordan forandringer i koden virker inn på tidsforbruket.

4.2.1 Valg av testbilder

Når forandringer er gjort med IDSL, er det viktig å teste hvordan forandringene påvirker gjenkjenningen. Derfor er det viktig med et gitt sett av bilder som testes hver gang. Ved å benytte de samme bildene hver gang, gir dette en indikasjon på hvordan forandringen påvirker gjenkjenningen. Etter at jeg hadde tatt 549 bilder, foretok jeg en vurdering av disse sammen med de 113 bildene Tørresen har tatt. Jeg har valgt å redusere det totale antall bilder som skal testes til 190. Ut fra disse er det valgt 107 bilder med fartsskilt i. Utvelgelseskriteriet var å finne bilder med stor spredning i tydelighet og vanskelighetsgrad når det gjelder gjenkjenning. Utvalget er så variert at det byr på store utfordringer å identifisere skilt og tyde dem. Noen av disse bildene er vist i figur 4.1. I utvalget er det også med fartsskilt jeg anser for vanskelig å finne. Det er bildet hvor ikke hele skiltet er innenfor bildet, figur 4.1 (b). I figur 4.1 (e) har fartsskiltet stått i sollys og blitt bleknet.

I tillegg til de 107 bildene *med* fartsskilt, er det plukket ut 83 bilder som *ikke* inneholder fartsskilt. Det er gjort for å teste at IDSL ikke feiltolker objekter som ikke er fartsskilt. Denne testen representerer situasjoner hvor IDSL ikke skal gi noe resultat. De fleste bildene i dette utvalget, er bilder av andre typer skilt, men det er også bilder hvor det ikke er skilt i det hele tatt. Noen av disse bildene er vist i figur 4.2.

4.2.2 Tidstest

For å måle tidsforbruket, blir det brukt tidtaker. Flere steder er det lagt inn "mellomtider" som måler tidsforbruket til de forskjellige fasene. Dette fører også til at det er lettere å se hvilke metoder som er kandidater til å bli implementert på en FPGA. Disse tidene, sammen med det totale tidsforbruket, forteller hvordan forandringer som gjøres påvirker hastigheten til IDSL.

Den første tiden som måles er hvor lang tid det tar å lese bildet fra fil. Det viser seg at denne tiden kan variere med opp til 60ms. En så stor variasjon i tidsforbruk kan overskygge forandringer i tidsforbruket til selve gjenkjenningen. I og med at dette er en del av IDSL som kan gjøres helt separat, samtidig som den ikke skal være med når bildene komm-

(a) Lett overskyet

(b) Ikke hele fartsskiltet

(c) Fartsskilt i skygge

(d) Tungt skydekke

(e) Sol, og bleknet fartsskilt

(f) Fartsskilt i sol

Figur 4.1: Utvalg av testbilder med fartsskilt

(a) To forbudsskilt

(b) Varselskilt

(c) Parkering forbudt skilt

(d) Forbikjøring forbudt skilt

(e) Maks tyngde skilt

Figur 4.2: Utvalg av testbilder uten fartsskilt.

er direkte fra kamera, har jeg valgt å utelate denne tiden fra det som inngår i gjennomsnittstiden. Jeg har valgt å måle tiden på fargefiltreringen (avsnitt **??**) og groingen av regioner (avsnitt 3.3.2) hver for seg. Dette er gjort fordi det er de delene av bearbeidingen som bruker mest tid. I tillegg er de klare kandidater til å bli implementert på FPGA. Lokalisering av skilt (avsnitt 3.4) og lokalisering og tyding av sifrene (avsnitt 3.5) er den siste tiden som måles. Grunnen til denne sammenslåingen er at de til sammen ikke utgjør mer enn 30% av gjenkjenningstiden. Det er dermed ikke den delen av IDSL det kan spares mest tid.

Testresultatene

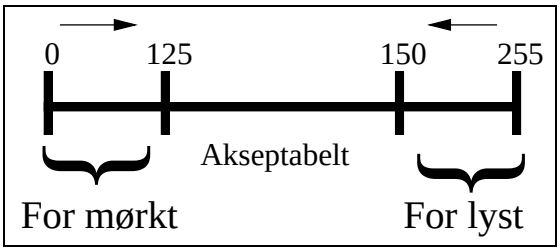
Resultatene av simuleringen er veldig viktig for utviklingen av IDSL. Det er dette som benyttes for å avgjøre hvilken innflytelse forandringer har på IDSL sin evne til å gjenkjenne fartsskilt. Under utviklingen er det sammenliknet resultater som er gjort før og etter forandringene. Når det skal testes tidsforbruk, er det viktig at IDSL har de samme forutsetningene både før og etter at forandringer er gjort. Derfor er tidsforbruket sammenliknet mellom to tester som kjøres etter hverandre, og dermed har identiske forhold. For selve gjenkjenningen av skiltene, har omgivelsene ingen betydning. Derfor kan gjenkjenningsresultater som er gjort på forskjellige tidspunkter sammenliknes.

4.3 Innlasting av bildet

Den endelige versjonen av IDSL skal få bildene fra et videokamera (med dette menes et digitalkamera som leverer et antall bilder per sekund). Det er derfor et krav at bildene som leveres har tilnærmet samme format uansett om de kommer fra kamera eller fil. Som nevnt i kapittel 3 velges hvert bilde manuelt i den originale versjonen av IDSL. En tilnærming til kameraløsningen er å gjøre det mulig å sjekke flere bilder etter hverandre.

4.3.1 Flere bilder på rad

Ved å gjøre det mulig å sjekke flere bilder etter hverandre, vil det simulere at det kommer flere bilder etter hverandre, som video. Samtidig vil testing av gjenkjenning kunne gjøres mer effektivt. Muligheten for å



Figur 4.3: Grafisk visning av intervallet for akseptable gjennomsnittsverdier

velge flere bilder som skal sjekkes er lagt til, og bildene legges i kø for å bli behandlet. IDSL begynner å behandle neste bilde etter at det forrige er ferdig behandlet.

Innlesningen av bilder tar lang tid i originalversjonen. Det tar ca 1,5 sekund å lese et bilde inn i IDSL. Som nevnt, baserte innlesingen seg på at RGB-verdiene til piksel hentes fra bildet som vises på skjermen. Metoden som henter pikselverdiene er derfor byttet ut. Den nye metoden legger verdiene for de forskjellige fargene inn i de respektive matrisene mer effektivt. Verdiene blir hentet fra *filen* som beskriver bildet og legges derfra inn i matrisene. Dette fører til at tiden IDSL bruker på å lese inn et bilde reduseres til ca 1/4 sekund. Dette fører til at simuleringer går vesentlig raskere.

4.3.2 Lysutjevning

I avsnitt 3.2 er det beskrevet at IDSL regner ut gjennomsnittlig pikselverdi, og bruker dette som estimat for å avgjøre om bilde er for mørkt eller ikke. De nye bildene som er tatt, viser at det også er nødvendig med en grense også for bilder som er for lyse. Ved å teste på forskjellige grenser, er det funnet ut at en fornuftig grense for bilder som er for lyse er gjennomsnittlig pikselverdi på 150. Grafisk visning av intervallet for akseptable gjennomsnittsverdier er vist i figur 4.3

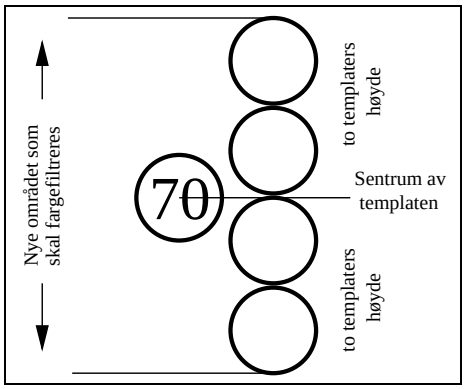
Som nevnt i avsnitt 3.2, justeres de bildene hvor gjennomsnittsverdien ligger utenfor det akseptable intervallet. Hver gang et bilde har gjennomsnittlig pikselverdi på utsiden av det akseptable intervallet, bruker den originale versjonen 40 ms på å justere pikselverdiene i hele bildet. For at bilder som ligger utenfor det akseptable intervallet også skal bli gjenkjent uten at dette tar vesentlig lenger tid, må metoden gjøres om.

Løsningen er at pikselverdiene forblir de samme, men grenseverdiene blir justert. Det betyr at *tre* verdier justeres istedenfor *alle* (307200) pikselverdiene i hele bildet. Ved å benytte denne metoden blir gjenkjenningsresultat det samme som tidligere, men tidsforbruket blir redusert. Dette fører til at gjenkjenningen går like fort, uavhengig av hvordan lysforholdene er i bildet.

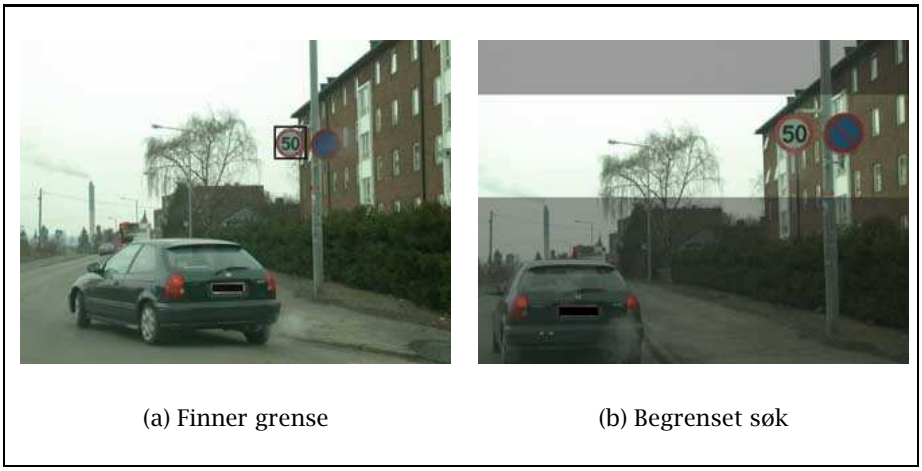
4.3.3 Sekvenser

I og med at flere bilder kommer etter hverandre er det mulig å benytte informasjonen om gjenkjenningen fra forrige bilde. For gjenkjenningsprosesser av denne typen er det to egenskaper som kan være nyttig for neste bilde. Den ene er skiltets plassering og den andre er størrelsen på skiltet. Jeg har valgt å benytte denne informasjonen til å begrense området hvor det skal gjøres fargefiltrering. Dette fører til at det ikke er nødvendig å filtrere hele bildet, noe som igjen reduserer antall punkter templatene sjekkes på. Det er forsøkt å avgrense søket til et kvadrat rundt området hvor forrige skilt ble funnet [14]. Dette viste seg ikke å være optimalt i og med at fartsskiltet kan bli skjult av gjenstander eller andre biler. Dersom dette inntreffer bør det være mulig å lokalisere et eventuelt skilt som befinner seg på den andre siden av veien. Det er derfor besluttet at fargefiltreringen skal foretas i *hele* bildets horisontale retning. Skilt på begge sider kan dermed bli funnet. Det er heller ikke noe problem med krappe svinger som ellers kan føre til at skilt forsvinner ut av det filtrerte området.

For den *vertikale* grensen, er det valgt å ta utgangspunkt i sentrum av det fartsskiltet som ble funnet i forrige bildet. Når det gjelder høyden på området, er det valgt å benytte størrelsen på forrige skilt som utgangspunkt. Grensen er satt til å være to ganger skiltets størrelse både oppover og nedover fra sentrum av forrige fartsskilts plassering. Dette er illustrert i figur 4.4. Skiltet vil være vesentlig mindre enn denne avgrensningen. På grunn av svinger, bakketopper ol. vil ikke nødvendigvis kameraet ha samme vinkel i forhold til veien for to etterfølgende bildene. Dumper og ujevnheter kan føre til at kameraet peker i en litt annen retning. Fargefiltrering på et bilde, som etterfølger et bilde hvor det er funnet fartsskilt, gjøres da bare innenfor det avgrensede området. En illustrasjon på denne begrensningen er vist i figur 4.5. Der blir skilt funnet i bilde (a), mens bilde (b) viser det området søket begrenses til i neste bildet. For å ta disse bildene er det brukt stillbildekamera som tar sekvenser hvor det er 750 ms mellom hvert bilde. Marginen vil derfor være vesentlig større når det benyttes videokamera.



Figur 4.4: Illustrasjon av vertikal avgrensning



(a) Finner grense (b) Begrenset søk

Figur 4.5: Illustrasjon på hvordan søket begrenses

En annen faktor som det også blir tatt hensyn til, er at skiltet vil være større desto kortere avstanden er mellom kameraet og skiltet. Derfor er det unødvendig å søke etter skilt som er mindre enn det som ble funnet i forrige bilde. Så for å lokalisere skilt, benyttes bare de templatene som er større eller like stor som den som passet på fartsskiltet i forrige bilde.

Dersom det blir funnet skilt i et bilde men ikke i det etterfølgende, vil metoden bli nullstilt. Det fører til at fargefiltreringen gjøres på hele neste bilde. Dette er en metode som egner seg godt når bildene kommer fra videokamera. I og med at systemet foreløpig ikke er ferdig utviklet består testbildene bare av enkeltsituasjoner for å teste egenskapene til systemet under forskjellige forhold. Metoden er derfor ikke benyttet etter at det

ble påvist at resultatene var gode. Det viste seg at behandlingstiden gikk ned. Området som skal fargefiltreres begrenses, som igjen begrenser stedene templater skal sjekkes. Når antall templater i tillegg er begrenset er det klart at tidsforbruket går ned. Reduksjon i tidsforbruket er avhengig av størrelsen på skiltet i bildet. Grunnen til dette er antall templater som skal sjekkes og størrelsen på området som skal filtreres er begge avhengig av størrelsen på skiltet.

4.4 Fargefiltrering

I avsnitt 3.3 forklares det hvordan fargefiltreringen gjøres.

4.4.1 Fargenes grenseverdier

Som nevnt er den opprinnelige versjonen av IDSL utviklet på grunnlag av et begrenset antall bilder. Alle er tatt i samme type lysforhold. Etter at nye bilder er hentet inn og testet av IDSL, viste det seg at grenseverdiene ikke er tilpasset alle lysforhold. Det er derfor forsøkt å justere grenseverdiene for å gjøre metoden mer robust. Forskjellige sammensetninger av grenseverdier er testet, og det er valgt å benytte den sammensetningen som førte til at flest fartsskilt blir funnet uten at tidsforbruket gikk vesentlig opp. Det nye kravet for at et piksel skal karakteriseres som hvitt, er at alle tre RGB-verdiene hver for seg må være større enn 108. For at et piksel skal karakteriseres som svart, må alle RGB-verdiene være mindre enn 122. Kriteriet for at et piksel er rødt, er at R-verdien må være større enn 77 og minst 17 større enn både G og B-verdiene:

- Pikselet er hvitt hvis: R > 108 og G > 108 og B > 108
- Pikselet er svart hvis: R < 122 og G < 122 og B < 122
- Pikselet er rødt hvis: R > 77 og G < R-17 og B < R-17

Grenseverdiene har en uønsket konsekvens. Ett og samme piksel kan karakteriseres som flere farger. Det viste seg at rødfarge som er litt bleknet både blir sett på som hvit og rødt. Dette fører til at enkelte skilt ikke blir funnet fordi hele eller deler av det røde på fartsskiltet blir karakterisert som hvitt istedenfor rødt. Skiltene som har litt blek rødfarge er enten gamle skilt eller skilt som har blitt utsatt for direkte sollys og derfor

bleknet med tiden. Det er ikke tilfredsstillende at disse fartsskiltene ikke blir funnet. Det er gjort forsøk på å finne grenseverdier for fargene som kan eliminere dette. Det viste seg at dette ikke er mulig uten at det går ut over andre situasjoner. Grenseverdiene ble forlatt og det ble vurdert om det finnes alternative måter å løse problemet på. Dette er forklart i avsnitt 4.4.2

4.4.2 Løsninger for enkelte spesialtilfeller

Fargeoverlapp

Siden enkelte av de røde pikslene feilaktig blir betraktet som hvite, er det laget en test for å sikre at rødt prioriteres. Samtidig som det letes etter 2x2 matrise som er hvit, blir det samtidig (parallellmetode) sjekket at disse pikslene ikke tilfredsstiller kravet for røde piksler. Dersom en av dem er røde vil ikke denne matrisen sees på som hvit. Dermed starter ikke det rekursive kallet, som er beskrevet i avsnitt 3.3. Dette fører til økning av tidsforbruket. Til gjengjeld øker gjenkjenningsprosenten vesentlig. Det finnes skilt hvor det røde er så bleknet at de ikke vil bli funnet selv med denne testen. Dette skjer når mesteparten av den originale rødfargen er borte.

En annen løsning som er testet, er å finne hele 2x2 matrisen med hvite piksler. Etter at denne er funnet sjekkes det om noen av disse tilfredsstiller kravet til rødt (sekvensiellmetode).

Disse metodene har lik gjenkjenningsprosent, men gjennomsnittlig tidsforbruk for den parallelle metoden er noe lavere enn den sekvensielle.

Gult

Som nevnt i kapittel 3, kan RGB-verdien for enkelte gulfarger oppfattes som rødt. Dette fører til problemer med enkelte bilder. Det gjelder bildene hvor det er forkjørsskilt under selve fartsskiltet. I slike situasjoner kan templater enkelte ganger legge seg delvis over forkjørsskiltet og delvis over fartsskiltet. For å unngå at dette skal inntreffe, og samtidig begrense antall steder templatet skal sjekke, bør de gule pikslene ikke filtreres som røde. Det er derfor lagt til et ekstra kriterium under kravet for at et piksel skal karakteriseres som rødt. Denne testen sørger for at gule piksler ikke blir tatt for å være røde. Kriteriet er at G-verdien

ikke skal være mer enn 50 større en B-verdien. Denne testen fører til en viss økning i gjenkjenningstiden. Det nye kriteriet for at et piksel skal karakteriseres som rødt er da:

- R > 77 og G < R-17 og B < R-17 og G-B < 50

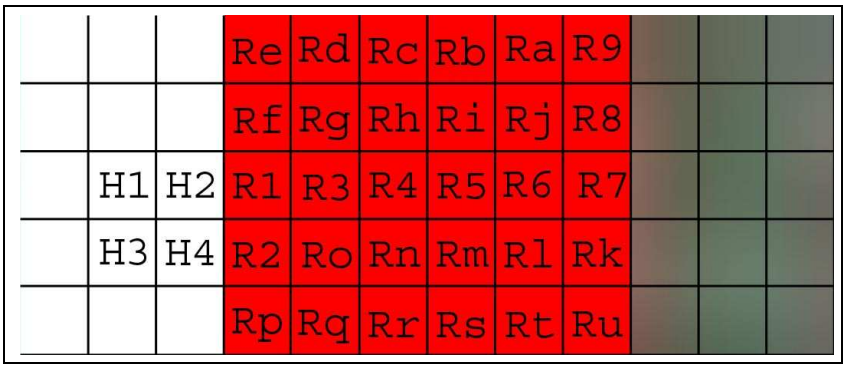
4.4.3 Segmentering

For å finne de røde områdene, blir det benyttet rekursivitet, som er forklart i avsnitt 3.3.2. For at det røde objektet skal lokaliseres, blir den samme testen gjentatt i veldig mange piksler. Dermed kan små endringer føre til stor forskjell i tidsforbruket. I den originale versjonen sjekkes alle retninger for alle pikslene. Dette vil si at pikselet som metoden fant i forrige analyse også blir sjekket av neste analyse. Det er derfor valgt å sende med informasjon om plasseringen til tidligere piksler som er sjekket. Den løsningen som er valgt å bruke, er å sende med informasjon om hvor det forrige pikselet som ble sjekket ligger i forhold til det pikselet som skal sjekkes. Dette fører til at alle pikslene som sjekkes, bare trenger å sjekke tre naboer. Dette er en fordel framfor den tidligere metoden som sjekket alle fire.

Som et forsøk på å gjøre det tydeligere, skal jeg bruke figur 4.6 som grunnlag for forklaringen. La oss si at det rekursive kallet er kommet til R4, og finner dette som rødt. Så går det videre til R5, som blir funnet til å være rødt. R5 vil i sin tur sjekke: R6, Ri, R4 og Rm. Men siden informasjon om hvor metoden var sist er tilgjenglig, blir ikke R4 sjekket. Dette fører til en reduksjon i antall tester, som igjen fører til reduksjon av tiden fargefiltreringen bruker.

Den nye rekursive metoden fører til en reduksjon i tidsforbruk på litt over 10% for fargefiltreringen, og litt under 10% for det totale tidsforbruket.

Det er også gjort forsøk der det sendes med informasjon om flere tidligere testede piksler. For at dette skal ha noen effekt, må minst tre steg bakover være tilgjengelig. Dette krever mange tester, og tidsforbruket for disse testene er større enn tiden som spares. Derfor blir bare informasjon om forrige piksel formidlet.



Figur 4.6: Liten bit av bilde for å forklare rekursivitet. H angir hvite piksler og R røde piksler

4.5 Lokalisere skilt

Det er i fasen med lokalisering av skilt det er mest tid å spare. I den originale versjonene av IDSL utgjorde lokalisering av skilt mellom 30% og 60 % av tiden IDSL bruker på å sjekke bilder.

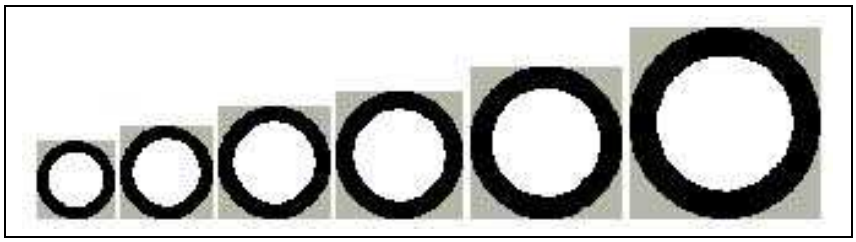
4.5.1 Templatene

Den aller største delen av tiden som går med til å lokalisere skilt blir brukt under templatsjekken.

Templatenes størrelse

Som nevnt i kapittel 3, passer enkelte templater ikke på noen av skiltene i testbildene. Derfor er fartsskiltenes størrelse i bildene analysert. Det viser seg at enkelte av templatene er så store at de aldri vil passe på noe bilde av et fartsskilt. Den maksimale størrelsen fartsskilt kan ha, vil aldri overskride 80x80 piksler med den oppløsningen som brukes på bildene.

Siden det er slått fast at det er en øvre grense for størrelsen på templatene, er det også undersøkt om det bør settes en nedre grense. Grunnen til at en slik grense er ønskelig, er at dersom skiltets størrelse i bilde er for liten vil det være vanskelig eller umulig å tolke sifrene inne i skiltes røde sirkel. Tester har vist at 32x32 piksler er den minste templatet som gjør det mulig å få pålitelig informasjon om skiltet. Dette har ført til at



Figur 4.7: Templater

antall templater er redusert fra de 10 som brukes i originalversjonen til 6. Det er viktig at alle skilt innenfor de definerte grensene blir funnet. Derfor må ikke størrelsesforskjellen mellom templatene være for stor. De 6 forskjellige templatene det er valgt å benytte har størrelse: 32x32, 38x38, 46x46, 52x52, 62x62 og 78x78. Disse templatene er vist i figur 4.7.

Etter å ha redusert antall og endret størrelsen på templatene, er tiden redusert til 10% av den tiden som den opprinnelige versjonen av metoden brukte på å lokalisere skilt.

Templatenes tykkelse

Som en del av prosessen med å forbedre gjenkjenningen, er det gjort forsøk som tester om endringer i tykkelsen på det røde i templatet fører til sikrere lokalisering av fartsskilt. Det er både gjort tester hvor tykkelsen er økt og redusert i forhold til det som finnes på fartsskilt. For testen med redusert tykkelse blir flere objekter som ikke er fartsskilt feilaktig tolket til å være det. Gjenkjenningen er ikke bedre, og det brukes mye tid på å forkaste forslag som ikke er fartsskilt. For templater som har økt tykkelse blir gjenkjenningen vesentlig dårligere. Derfor benyttes templater hvor den røde ringen er like stor som den som finnes i skilt.

4.5.2 Testing av templatets plassering

Som nevnt i kapittel 3, sendes enkelte objekter fra templatsjekk videre i gjenkjenningsprosessen uten at de er fartsskilt. For at dette ikke skal inntreffe, er det sett nærmere på testene som gjøres hver gang et templat stemmer tilstrekkelig. Området disse testene gjøres på er også analysert. Det er forsøkt å finne et området disse testene kan gjøres på som med

sikkerhet kan bekrefte eller avkrefte om templatet ligger over fartsskilt.

Testene

Templatene blir ført over rødmatrisen for å sjekke hvordan de passer med objektene. I den forbindelse foretas det mange sammenlikninger. Det er derfor viktig at man forkaster forsøk så tidlig som mulig i prosessen, dersom det er klart at det ikke kan være fartsskilt. Som forklart i avsnitt 3.4.2, gjøres tre nye tester for å analyse objektet templatet er plassert over. Det viser seg at disse testen ikke er tilstrekkelig til å forkaste alle objekter som ikke er skilt. Prosessen er derfor *utvidet* med to tester. En av testene gjøres i sentrum av templatet, mens den andre gjøres i venstre kant.

Den nye testen i sentrum gjøres for å forsikre om at det finnes både svarte og hvite piksler der. De svarte pikslene som blir funnet skal tilhøre sifrene som står i skiltet, og de hvite skal være mellomrommet mellom dem. Dersom ingen eller bare den ene fargen blir funnet, vil forslaget til skilt forkastes. Med en slik test vil skilt som forbud mot alle kjøretøyer og innkjøring forbudt forkastet. Andre objekter som har likhetstrekk med fartsskilt kan også bli forkastet av denne testen.

Den siste testen foretas for å være sikker på at det er røde piksler i venstre kant av templatet. Blir det ikke funnet røde piksler, forkastes templa. Som nevnt i avsnitt 3.5.2, kan metoden som fjerner alt annet enn sifrene blanke ut linjer i skiltet. Test av piksler i venstre kant er en av forandringene som er gjort for å forhindre dette. Den andre forklares i avsnitt 4.6.2. I avsnitt 4.4.2 diskuteres problemet med at templatet legger seg delvis over fartsskilt og delvis over forkjørsskilt. Testen av piksler i venstre kant eliminerer dette problemet. Det er derfor valgt å utelate kriteriet som eliminerer gult fra kravet for at piksel skal karakteriseres som rødt. Dermed benyttes kriteriene som er nevnt i avsnitt 4.4.1 for å bestemme om et piksel er rødt. Området hvor det må finnes røde piksler er vist som den hvite firkanten på den røde sirkelen i figur 4.8. De nye testene, som benyttes sammen med de tre opprinnelige, er altså:

- Finnes svarte og hvite piksler inne i skiltet.
- Finnes røde piksler til venstre i templatet.

Figur 4.8: Områder hvor testene utføres

Området for testene

For å være sikker på å forkaste de skiltene som skal forkastes, viser det seg at det ikke er nødvendig å sjekke hele området innenfor den røde sirkelen. Det er derfor gjort analyse av skiltet med tanke på hvor testene som er beskrevet over kan gjøres mest mulig sikre og tidseffektive. Forskjellige områder og størrelser er testet. Testene viser at området er identifisert til 1/6 av skiltets størrelse i horisontal retning med halvparten på hver side av sentrum. I vertikal retning blir bare 3 piksellinjer sjekket. Ved å søke i dette området er det garantert å finne både sentrum av skiltet og litt utenfor. På denne måten vil testene som gjøres foreta en sikker bekreftelse på om ønskede egenskaper finnes eller ikke, uansett hvordan templatet ligger i forhold til fartsskiltet. Testene vil også bidra til en sikrere eliminering av objekter som ikke er skilt. Det inverterte området i figur 4.8 illustrerer hvor testene gjøres.

Hovedmålet med forandringene som er gjort er å unngå at "falske" fartsskilt blir sendt videre i gjenkjenningsprosessen. Tester fører til at vesentlig flere "falske" fartsskilt ikke blir sendt videre og gjenkjenningen blir dermed bedre. Tester avdekket at det nye området som benyttes for å teste templatets plassering, sammen med de nye testene, har ført til at tidsforbruket reduseres med 25% for lokalisering av skilt.

Med tester er det gjort forsøk som skal sørge for at templatet dekker hele skiltet. Dette er gjort ved at templatet blir forkastet dersom ikke det er rødt piksel helt på topp, bunn, høyre og venstre av templatet. Dette er en usikker test. Det finnes situasjoner der skiltet står i vinkel i forhold til kamera eller deler av skiltet er dekket. Som tidligere nevnt, ligger også mange av templatene litt skjevt på skiltet fordi det er litt sprang i størrelsen mellom templatene som testes. Dermed forkastes

disse fartsskiltene feilaktig. Derfor blir det bare testet at det finnes røde piksler i venstre kant.

4.6 Lokalisering og tyde sifrene

Den raskeste fasen av IDSL er å finne sifrene i skiltet. Systemet bruker under 10 ms på denne operasjonen.

4.6.1 Klargjøre området

Som nevnt i avsnitt 3.5.1 viser det seg at hele den røde ringen på skiltet i enkelte bilder ikke blir sett på som rød. Som nevnt er det viktig at områdene er så homogene som mulig for å få best mulig resultat. For bilder som har lite lys, er problemet at deler av sirkelen oppfattes som svart. Problemet med dette, er at metoden kan se på disse punktene som del av sifrene. Dette kan også føre til at metoden som skal trekke ut sifrene, og som er beskrevet i avsnitt 3.5.2, får problemer med å identifisere sifrene. Denne metoden kan feiltolke slik at kanten på skiltet aldri blir funnet. På grunn av dette vil hele linjer blankes ut.

For å unngå at linjer blankes ut fordi sirkelen er svart, gjøre de pikslene som er svarte, som helt klart ikke er sifrene og har kontakt med den røde sirkelen, om til rødt. Metoden som fremhever sifrene vil da lettere skille det som er sifre fra resten av fartsskiltet.

4.6.2 Trekke ut sifrene

Testene som er beskrevet i avsnitt 3.5.2, er som tidligere nevnt, så strenge at deler av enkelte sifrene fjernes istedenfor å bli ekstrahert. Kravet for at piksel skal være del av sifrene er derfor gjort om noe. Som nevnt i avsnitt 3.5.2, må det være røde piksler på en linje før svarte piksler kan tilhøre sifrene. Siden det gjøres test på at det må være rødt i venstre kant av templatet, er dette krav opprettholdt. Det er lagt til test som sikrer at skilt ikke kan begynne til venstre for midten av templatet. Derimot er det ikke lenger nødvendig at det er rødt etter at det er funnet piksler som tilhører sifrene. Dersom det kommer rødt lenger til høyre enn midtveis i templatet, betraktes dette som høyre kant av skiltet, og svarte piksler som etterfølger dette betraktes ikke som del av sifrene.

Krav for at svart piksel skal tilhøre sifrene:

1. Det må finnes rødt piksel til venstre for midten av templatet
2. Etter at rødt piksel er funnet på en linje, vil svarte piksler som blir funnet på samme linje betraktes som del av sifrene.
3. Dersom det kommer rødt piksel til høyre for midten av templatet, vil ikke flere svarte piksler i den raden registreres som del av sifrene.

4.6.3 Størrelse og plassering av sifrene

Denne metoden finner grensene rundt sifrene og er forklart i avsnitt 3.5.3. I utgangspunktet blir disse grensene funnet ved at man går fra kanten av templatet og søker seg innover til man kommer til sifrene. Dette fører til at metoden er følsom for støy i bildet utenfor sifrene. For å redusere denne feilkilden er det gjort forsøk der tilnærmingen av sifrene startet noe nærmere sentrum enn kantene av templatet. Søket starter 1/8 av templatets størrelse nærmere sentrum fra templatets ytterkant. Dette forbedret tilnærmingen, men for enkelte fartsskilt hvor templatet ikke dekket skiltet ordentlig, blir deler av sifrene kappet av. Det er derfor foretatt en test der metoden er snudd slik at den tester fra sentrum og ut. Grensen til sifrene er da den første linjen som blir funnet uten at den inneholder svarte piksler. Dette fører til at grensen rundt sifrene er tettere, noe som igjen gjør at sifrene blir tydeligere i egenskapsmatrisen.

Det gjøres enda en test for å være enda sikrere på at det er fartsskilt som er funnet. Det blir testet at bredde og høyde av sifrene er minst 1/3 av templatets størrelse. Dersom det ikke er det, blir forslaget forkastet som fartsskilt. Hvis sifrene som er funnet er større en 1/3 av templatets størrelse, blir sifrene ekstrahert.

Det er gjort forsøk på å skille sifrene på en mer elegant måte. Kriteriet som må tilfredstilles for å skille sifrene, er at det finnes en vertikal pikselrad som bare inneholder hvite piksler. Dette er en metode som fungerer god for de fleste skiltene. Problemet er 70 skilt, der spissen på 7 tallet og 0'en som oftest deler minst en pikselrad. Derfor er det valgt å beholde den originale måten å skille sifrene på.

4.6.4 Tyde sifferet

Som nevnt i avsnitt 3.5.4 er det bare mulig å gjenkjenne 50 og 60 skilt i den originale versjonen av IDSL. For å gjøre det mulig å gjenkjenne flere typer fartsskilt er det laget egenskaper for flere typer tall. Det viser seg at det ikke er opplagt hva som skiller egenskapene for de forskjellige tallene. Gjenkjenningsprosenten for sifrene er relativt lav.

Matrisene for sifrene som genereres i IDSL ble gitt til Tørresen. Han laget tester som sjekket hvordan nevralt nett og evolusjonær maskinvare egner seg til gjenkjenning av skilt. Resultatet av disse testene viste seg å være gode. 94,5% av numrene blir korrekt klassifisert når evolusjonær maskinvare ble benyttet [27]. For tester med nevrale nett viste det seg at 100% av sifrene ble korrekt klassifisert [26].

4.7 Simuleringsresultater

Alle simuleringsresultatene som beskrives er hentet fra simuleringer jeg har gjort på min datamaskinen. Maskin har en Atlon AMD 1600MHz prosessor og 512MB RAM (Random Access memory). I dette avsnittet skal jeg sammenlikne den originale versjonen av IDSL med den nye versjonen jeg har utviklet. Sammenlikningene blir gjort både med tanke på gjenkjenningsresultat og tidsforbruk.

IDSL blir testet med to ulike tester, og det er de samme som blir brukt hver gang. Den ene testen er med 107 bilder som inneholder fartsskilt, mens den andre er 83 bilder som *ikke* inneholder fartsskilt. Disse to settene med bilder er beskrevet i avsnitt 4.2.1.

4.7.1 Gjenkjenningsresultat

Først skal jeg beskrive gjenkjenningsresultatet for den originale versjonen av IDSL. Sifrene som er trukket ut er ikke forsøkt tydet i nevralt nett, slik som den endelige versjonen. Derfor anses de skiltene som det blir funnet en eller flere punkter i den normaliserte utgaven av sifferet (7x5 matrisen) for å være korrekt klassifisert. Dette tatt i betraktning blir det korrekt klassifisert 39 av de 107 bildene som inneholder fartsskilt. Dette vil si at 36,5% av skiltene blir korrekt klassifisert. For testen som ikke inneholder fartsskilt, lokaliserer IDSL 24 fartsskilt. Dette vil si 71% korrekt klassifikasjon. Totalt blir 51,5% av bildene korrekt klassifisert.

	Original versjon	Ny versjon
Test med fartsskilt	36%	85,9%
Test uten fartsskilt	71%	95,1%
Totalt	51,5%	90%

Tabell 4.1: Oppsummering av gjenkjenningsresultatene

Gjenkjenning av ekstraherte siffer er for den endelige versjonen er testet i et nevralt nett. Det viser seg at alle de ekstraherte sifrene blir korrekt klassifisert. Det betyr at for alle skilt som klassifiseres som fartsskilt, blir sifferet som angir fartsgrensen korrekt gjenkjent. For testen med bilder som inneholder fartsskilt blir det korrekt lokalisert 92 av de 107 skiltene. Dette vil si 85,9% korrekt klassifikasjon. I bildene som ikke inneholder fartsskilt, blir det lokalisert fartsskilt i 4 av de 83 bildene. Dette vil si at 95,1% av bildene blir korrekt klassifisert. Totalt blir 90% av bildene korrekt klassifisert. I tabell 4.1 er gjenkjenningsresultatene oppsummert. Resultatene viser at potensialet for forbedring er størst med tanke på å lokalisere fartsskilt i bilder som inneholder fartsskilt.

4.7.2 Tidsforbruk

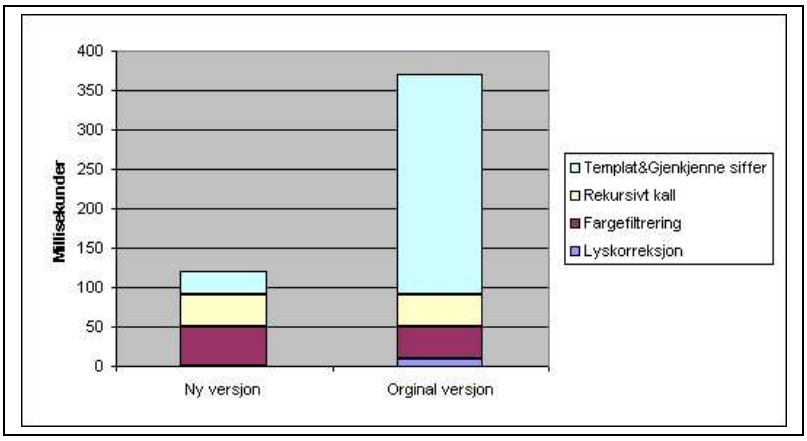
For at alle resultatene skal være så nøyaktig som mulig er det valgt å gjøre alle testene samtidig, slik at de har samme forutsetning.

Totalt tidsforbruk

Sammenlikning av tidsforbruket mellom den originale versjonen og den endelige versjonen av IDSL er vist i figur 4.9. Stolpen til venstre viser gjennomsnittlig tidsforbruk for den nye versjonen. Stolpen til høyre viser gjennomsnittet til den originale versjonen. Stolpene er delt inn i forskjellige faser av gjenkjenningsprosessen.

Den originale versjonen av IDSL bruker i gjennomsnitt 443,2 ms på å sjekke et bilde som inneholder fartsskilt. For bilder som ikke inneholder fartsskilt bruker den originale versjonen 328 ms i gjennomsnitt.

I den nye versjonen av IDSL er tidsforbruket i gjennomsnitt 121 ms for å sjekke et bilde som inneholder fartsskilt. For bilder som ikke inneholder fartsskilt er gjennomsnittlig tidsforbruk på 123,7 ms. I flere av



Figur 4.9: Sammenlikning av tidsforbruk for den originale og den nye versjonen av IDSL

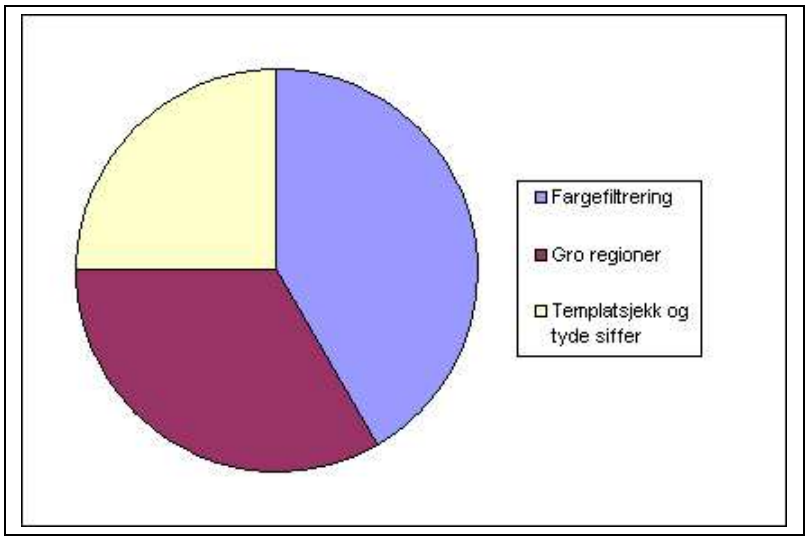
bildene som ikke inneholder fartsskilt er det store mengder piksler som tilfredsstiller kravet til rødt. Mengden av røde piksler har stor innflytelse på tidsforbruket. Dette er grunnen til at gjennomsnittlig tidsforbruk for bildene uten fartsskilt er høyere enn for bilder som inneholder fartsskilt.

Metodenes tidsforbruk

Testene det vises til i dette avsnittet, tar utgangspunkt i den endelige versjonen av IDSL. For å gjøre det enkelt å se hvordan hovedforandringene som er gjort i IDSL påvirker gjenkjenningen og tidsforbruket, er metoden som testes i hvert tilfelle det eneste som er forandret i forhold til den *nye* versjonen av IDSL.

Fordelingen av tidsforbruket for de forskjellige fasene i den nye versjonen av IDSL er vist i figur 4.10.

Tabell 4.2 viser resultatene av de forskjellige testene som er gjort for å analysere de forskjellige metodene. Det vises både gjenkjenningsresultat og tidsforbruk. Tidsforbruket er også vist i figur 4.11. Der viser de forskjellige stolpene hvor mye mer tid den originale versjonen av metoden bruker i forhold til de nye versjonen av IDSL. Det er to søyler for hver metode. Den ene viser hvor mye mindre tid som blir brukt for bilder som inneholder fartsskilt. Den andre søylen viser hvor mye mindre tid som blir brukt for å sjekke bilder som ikke inneholder fartsskilt. I figuren er betegnelsen for metoden lik den som er beskrevet i tabellen. I



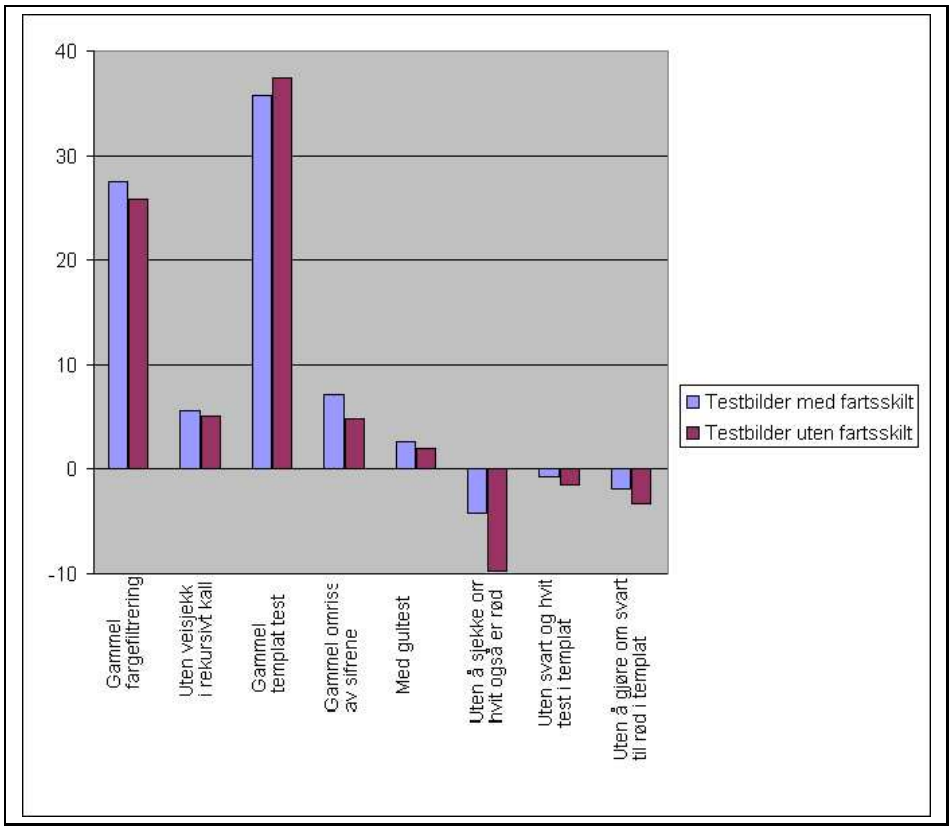
Figur 4.10: Diagram over tidsforbruk for de forskjellige fasene av den nye IDSL versjonen

tabellen blir det henvist til avsnitt, som beskriver metoden.

Under sluttføringen av denne oppgaven er det også gjort ytterligere forbedring av IDSL. Denne forbedringen er beskrevet i avsnitt 4.4.2. Forbedringen er gjort så sent at tiden ikke har tillatt å gjøre alle simuleringene en gang til med denne forbedringen. Resultatet av de simuleringene som er gjort viser at tidsforbruket går ned med litt over 6 ms. Det vil si at det nye gjennomsnittet for å sjekke et bilde er ca 116 ms for bilder med fartsskilt og ca 119,7 ms for bilder uten fartsskilt.

Tabell 4.2: Simuleringsresultater

Metoder som testes	Test type	Gjennomsnittlig tidsforbruk pr bilde	Antall skilt funnet	% korrekt klassifisering
Endelig versjon	Med skilt	121ms	92	85,9
	Uten skilt	123,7ms	4	95,1
Gammel fargefiltrering (avsnitt 3.3)	Med skilt	148,5ms	87	81,3
	Uten skilt	149,5ms	4	95,1
Uten veisjekk i rekursivt kall (avsnitt 3.3.2)	Med skilt	126,6ms	92	85,9
	Uten skilt	127,7ms	4	95,1
Gammel templat test (avsnitt 3.4.1)	Med skilt	156,7ms	86	80,3
	Uten skilt	161,2ms	5	93,9
Gammel omriss av sifrene (avsnitt 3.5.3)	Med skilt	128,1ms	88	82,2
	Uten skilt	127,5ms	6	92,8
Med gultest (avsnitt 4.4.2)	Med skilt	123.6ms	92	85,9
	Uten skilt	125,7ms	4	95,1
Uten å sjekke om hvit også er rød (avsnitt 4.4.2)	Med skilt	116,7ms	82	76,6
	Uten skilt	113,9ms	4	95,1
Uten svart og hvit test i templat (avsnitt 4.5.2)	Med skilt	120,2ms	84	78,5
	Uten skilt	122,1ms	5	93,9
Uten å gjøre om svart til rød i templat (avsnitt 4.6.1)	Med skilt	119,1ms	91	85
	Uten skilt	120,4ms	4	95,1



Figur 4.11: Oversikt over hvor mye lenger tid de gamle metodene bruker i forhold til de nye. Negativ verdi betyr at metodene øker gjenkjennings-tiden

Kapittel 5

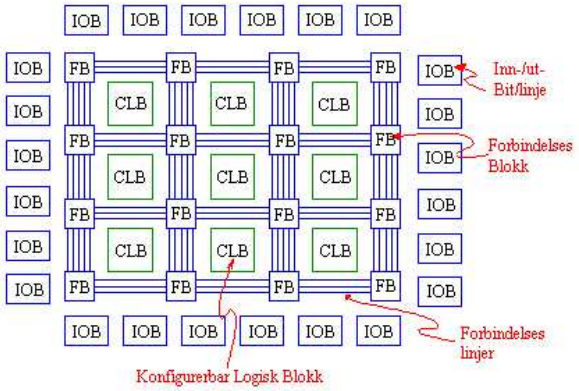
FPGA tilnærming og implementasjon

I dette kapitlet blir det gitt en kort forklaring på hva en Field Programmable Gate Array (FPGA) er og hvordan den er bygget opp. Det gis begrunnelse for hvorfor det er ønskelig å gjøre enkelte deler av gjenkjenningen på en FPGA. Det vil også bli gjort rede for testene som er gjort for å finne ut om IDSL kan ha fordel av å benytte FPGA, og resultatene av disse testene. Til slutt vil det bli gjort rede for hvordan implementeringen er gjort og hvor fort det kan forventes at gjenkjenningen skal gå.

Skiltgjenkjenningssystemer som benytter rekonfigurerbar logikk er ikke kjent fra litteraturen. Derimot er det flere som har benyttet slik logikk til bildebehandling [3, 5, 16].

5.1 Hva er en FPGA?

En Field Programmable Gate Array (FPGA) er en rekonfigurerbar logisk komponent. Det vil si en komponent man kan konfigurere til å gjøre logiske funksjoner gjentatte ganger etter at komponenten er produsert.



(a) FPGA [29] (b) Innsiden av FPGA [12]

Figur 5.1: FPGA

5.1.1 Oppbygning av FPGA

FPGA-teknologien har utviklet seg enormt de siste årene og blitt meget fleksible. En FPGA er vist i figur 5.1 (a), mens en forenklet beskrivelse av innsiden er vist i figur 5.1 (b). Informasjon inn og ut av FPGAen går gjennom Inn/Ut- bit/linje (IOB). Hensikten med disse blokkene er gjøre det mulig for FPGAen å kommunisere med eksternt utstyr. Blokkene kan konfigureres til å støtte flere forskjellige informasjonsutvekslingsmåter. All behandling av informasjon skjer i konfigurerbare logiske blokker (CLB) som er hovedbestanddelen i FPGAer. CLBene er plassert rundt på hele FPGAen, og det kan være opp til 125 000 CLBer på en FPGA [30]. Hver CLB kan programmeres til å utføre en begrenset logisk funksjon, men ved å koble flere sammen kan operasjonene bli komplekse. CLBene er koblet sammen med forbindelses linjer, disse kan sees på som ledninger. Disse ledningene overfører informasjon mellom CLBene, og ligger horisontalt og vertikalt på FPGAen. For å få informasjonen til riktig CLB blir forbindelsene koblet sammen i forbindelsesblokker (FB). Hvis et signal skal fra en CLBene og ut av FPGAen, blir det sendt over forbindelsen til IOBene, som igjen sender det ut av FPGAen til eksterne komponenter.

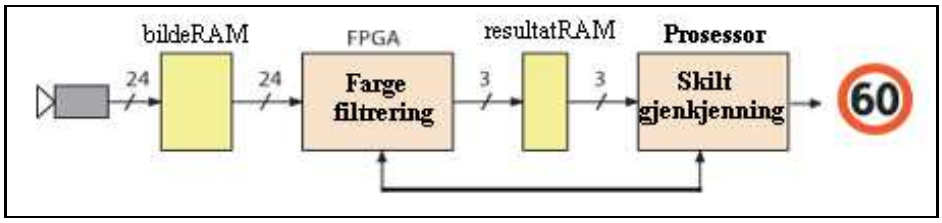
5.1.2 Konfigurasjon av FPGA

Konfigurasjon av FPGA kan gjøres på "CLB-nivå", som vil si at man manuelt konfigurerer innholdet i alle CLBene, og alle forbindelser mellom disse. Dette blir fort svært uoversiktlig, og gjøres aldri i praksis. Det blir istedenfor beskrevet hvordan FPGAen skal oppføre seg i maskinvarebeskrivende språk (for eksempel VHDL (Very High Speed Integrated Circuits Hardware Description Language) eller Verilog). Beskrivelsen blir syntetisert til en gyldig FPGA-konfigurasjon av et synteseverktøy. Syntetiseringen skjer automatisk, men konstruktøren kan gi beskjed til synteseverktøyet at konfigurasjonen skal gjøres med hensyn på plass, hastighet e.l.

5.2 Hvorfor FPGA

En del av denne oppgaven går ut på å lage oppsett som gjør IDSL klar for prototyptesting. Det er stilt krav om at prototypen skal sjekke minst 10 bilder per sekund. Som nevnt i avsnitt 4.7 er gjennomsnittlig tidsforbruk 121 ms per bilde som inneholder fartsskilt. Dette tilsvarer at 8 bilder blir sjekket per sekundet. Dermed må tidsforbruket ytterligere ned for å tilfredsstille tidskravet. Det er jobbet mye med å optimalisere IDSL, og det er ikke mulig å få det til å gå 21 ms raskere uten at dette går ut over kvaliteten og sikkerheten til gjenkjenningen. Det er derfor valgt å forsøke å implementere en fase av IDSL på FPGA, for å teste ut om dette fører til at tidsforbruket reduseres tilstrekkelig.

Et forslag til oppsett av IDSL sin prototyp er vist i figur 5.2. Det er valgt å forsøke å implementere den fasen av IDSL som gjør fargefiltreringen på FPGA. Det er denne delen av IDSL som bruker lengst tid. Fargefiltreringen bruker mellom 30% og 40% av det totale tidsforbruket. Testdata og resultater for denne delen er lett tilgjengelig, noe som gjør dette til en gunstig del å teste. Bildeoperasjoner går mellom 8 og 100 ganger så fort på en FPGA som på en PC med 800 MHz Pentium III prosessor [3]. Ved å benytte denne informasjonen, er det mulig å få en antydning om hvilken ytelse som kan være oppnåelig. Fargefiltrering som gjøres med IDSL koden tar omtrent 50 ms. Maskinen som IDSL koden er testet på er dobbelt så rask som en PC med 800 MHz Pentium III prosessor. Dette tilsier at fargefiltrering på FPGA vil da anslagsvis bruke maksimalt 1/4 av tiden IDSL koden bruker (50ms/4=12,5ms). Hele gjenkjenningsprosessen tar da 93,5 ms noe som tilsvarer 10,8 bilder per sekund.



Figur 5.2: Forslag til oppsettet for prototyptest [25]

5.3 Sjekk før FPGA implementasjon

Som tidligere nevnt er det valgt å teste fargefiltrering på FPGA. Når en metode skal gjøres på FPGA fremfor å gå på en prosessor, krever dette ofte enkelte forandringer eller modifiseringer av selve metoden. Dette var også tilfellet for fargefiltreringen. Derfor er det nødvendig å teste om metoden som skal implementeres på FPGA også kommer frem til samme resultat som tidligere. Som nevnt i avsnitt 3.3, gjøres fargefiltreringen i IDSL ved å sjekke om hvite piksler har røde området som nabo. Derimot filtreres ikke svarte piksler før templatene skal sjekkes. Søket etter svarte piksler begrenses til inne i templatene.

I en FPGA er det begrenset mengde med plass til å lagre verdier. Det er derfor nødvendig å lagre bildet på utsiden, som er beskrevet i avsnitt 5.4. Det tar tid å flytte informasjon inn i en FPGA, derfor er det ikke mest optimalt å søke etter 2x2 matrise med hvite piksler før det sjekkes om naboene er røde. Løsningen som er valgt å benytte, er å filtrere hele bilde med hensyn på både rødt og hvitt. Når dette er gjort er informasjonen i bildet redusert til det som er interessant. Dermed kan denne informasjonen få plass inne i FPGAen. Dette fører til at det går vesentlig raskere å finne de røde regionene som har hvit matrise som nabo.

IDSL sjekker ikke svarte piksler før templater sjekkes. I og med at originalbildet ikke vil være tilgjenglig senere i gjenkjenningsprosessen, er det nødvendig at svarte piksler også er tilgjengelig. Det betyr at det er tre farger som filtreres på FPGAen, rød, hvit og svart. Resten av IDSL må forholde seg til denne informasjonen.

For å teste den nye fargefiltreringen er det valgt å representere hver farge ved å gi det et tall som legges inn i en resultatsmatrise (Rød=1, Hvit=2, Svart=4). Hvis et piksel tilfredstilte kravet til flere farger, blir tallene lagt sammen (Rød og Svart=5, Rød og Hvit=3, Alle=7). Tallene er valgt slik at det er entydig hvilke farger som er funnet. Resten av IDSL er gjort

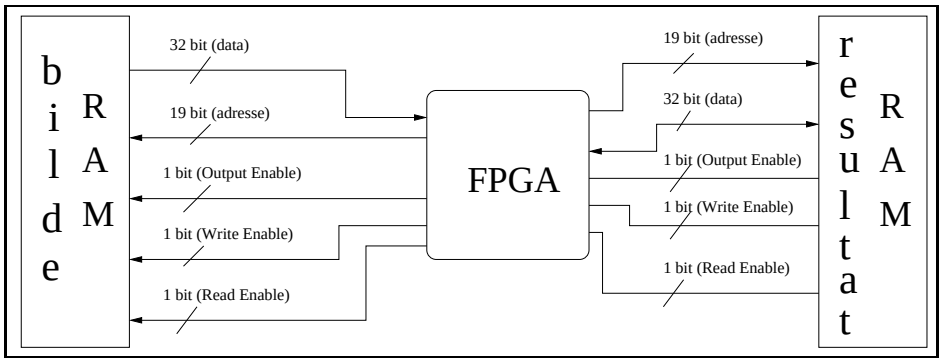
om, slik at den sjekker fargen på piksler ut fra denne matrisen. Den nye metoden er testet, og den viste at gjenkjenningsresultatet forble det samme. Det viste seg til og med at gjennomsnittlig tidsforbruk per bilde gikk ned med 1 ms. Dette betyr at denne måten å filtrere på gir like godt resultat som den som er beskrevet i avsnitt 4.4, og kan benyttes på en FPGA.

5.4 Oppsett for testen

Den nye fargefiltreringen er skrevet i VHDL, og denne metoden er testet ved å simulere den separat fra resten av IDSL. Simuleringsverktøyet som ble benyttet er Modelsim. Som nevnt er forslag til oppsett for prototyptesten vist i figur 5.2. Her blir det beskrevet hvordan systemet rundt FPGAen testes. Når simuleringen starter, er bildeRAM fylt med et bilde. Så hentes ett og ett piksel inn i FPGA'en, hvor det filtreres. Resultatet skrives til resultatRAM, samtidig som FPGAen ber om neste piksel fra bildeRAM. Når dette er gjort for hele bildet, blir innholdet i resultatRAM skrevet til en fil. Denne filen benyttes for å sjekke at resultatet for versjonen som og VHDL-versjonen er like.

Bildene IDSL bearbeider er forholdsvis store og krever derfor mye plass. Skal det benyttes en FPGA som har lagringskapasitet til å lagre hele bildet, fører det til at en uhensiktsmessig stor og dyr FPGA må benyttes. Derfor er det valgt å benytte ekstern RAM hvor bildet kan aksesseres fra FPGAen. Dette fører til at en langt mindre og dermed billigere FPGA kan benyttes. Det er valgt å benytte to RAMer i oppsettet. Den ene skal benyttes til å lagre hele bildet (bildeRAM), mens den andre skal benyttes til å lagre resultatet (resultatRAM). Oppsettet er vist i figur 5.3, og hele oppsettet til prototypen er vist i figur 5.2.

Det er valgt å bruke en SRAM (static random access memory) modell [31] som er skrevet i VHDL. Denne modellen ble funnet på Internet, og er utviklet av Andre Klindworth som er ansatt på Universitetet i Hamburg. Dette muliggjør simulering av systemet og interaksjon med RAM uten å ha et fysisk system til stede. Modellen inneholder variable som kan endres, slik at modellen passet til flere typer SRAM. Det er valgt å ta utgangspunkt i en Micron 16Mb SDRAM (MT2LSDT432UG-8). Verdiene i VHDL modellen er modifisert, slik at modellen oppførte seg på samme måte som RAMen som er valgt å bruke. Dermed er det mulig å simulere systemet realistisk. Hver posisjon i RAMen inneholder 32 bit og siden det trengs 8 bit for hver farge, vil det si at det er nødvendig med 24 (3·8) bit for å representere ett piksel. Dermed er det plass til mer enn et pik-



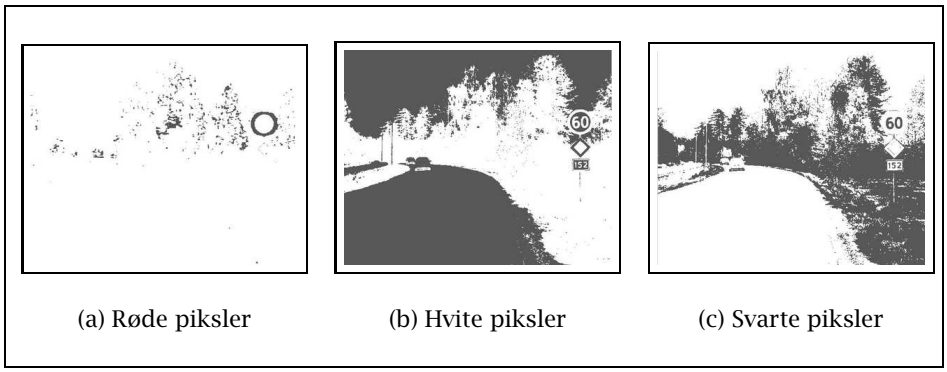
Figur 5.3: Hvordan oppsettet av FPGAen ser ut

sel i hver adresserbare lokasjon. For enkelhets skyld er det valgt å ikke benytte de siste plassene, slik at hver lokasjon i bildeRAM inneholder ett piksel. Resultatet av fargefiltreringen er på 3 bit og også i resultatRAM er det valgt å benytte en posisjon per piksel. Synkronisering av RAM-akses til henholdsvis kamera og prosessor er avhengig av løsningen som velges for disse. Siden dette ikke er en del av denne oppgaven, er ikke synkroniseringen vurdert her.

5.5 Fargefiltrering på FPGA

Et blokkdiagram av hvordan fargefiltreringen gjøres i FPGA er vist i figur 5.5. Prosessen med å fargefiltrere et bilde begynner ved at det gis beskjed til bildeRAM hvilken pikselverdi den skal sende. Verdien blir lest inn i FPGAen hvor de forskjellige fargeverdiene (RGB) blir separert. Disse verdiene blir lagt sammen med pikselverdiene for de foregående pikselverdiene som et ledd i utregningen av gjennomsnittlig pikselverdi for bildet. Så gjøres selve filtreringen ved å sjekke om fargen på pikselet tilfredsstiller kravene for de forskjellige fargene som er beskrevet i avsnitt 4.4.1. Resultatet fra fargefiltreringen er en bitvektor, hvor hver av de tre fargene (rød, hvit, svart) har sin faste plass. Disse plassene blir satt til 1 hvis kravet for fargen bitet representerer er oppfylt. Når pikselet er filtrert, blir resultatet skrevet til resultatRAM. Samtidig som resultatet skrives, leses verdien til neste piksel fra bildeRAM. Resultat av fargefiltreringen er vist i figur 5.4. Hver av fargene har hvert sitt bilde for å vise hvordan de filtrerte bildene for hver av fargene ser ut.

Når hele bildet er filtrert og resultatet skrevet til resultatRAM, blir summen av verdiene til alle pikslene sendt til resultatRAM. Dette gjøres fordi



(a) Røde piksler (b) Hvite piksler (c) Svarte piksler

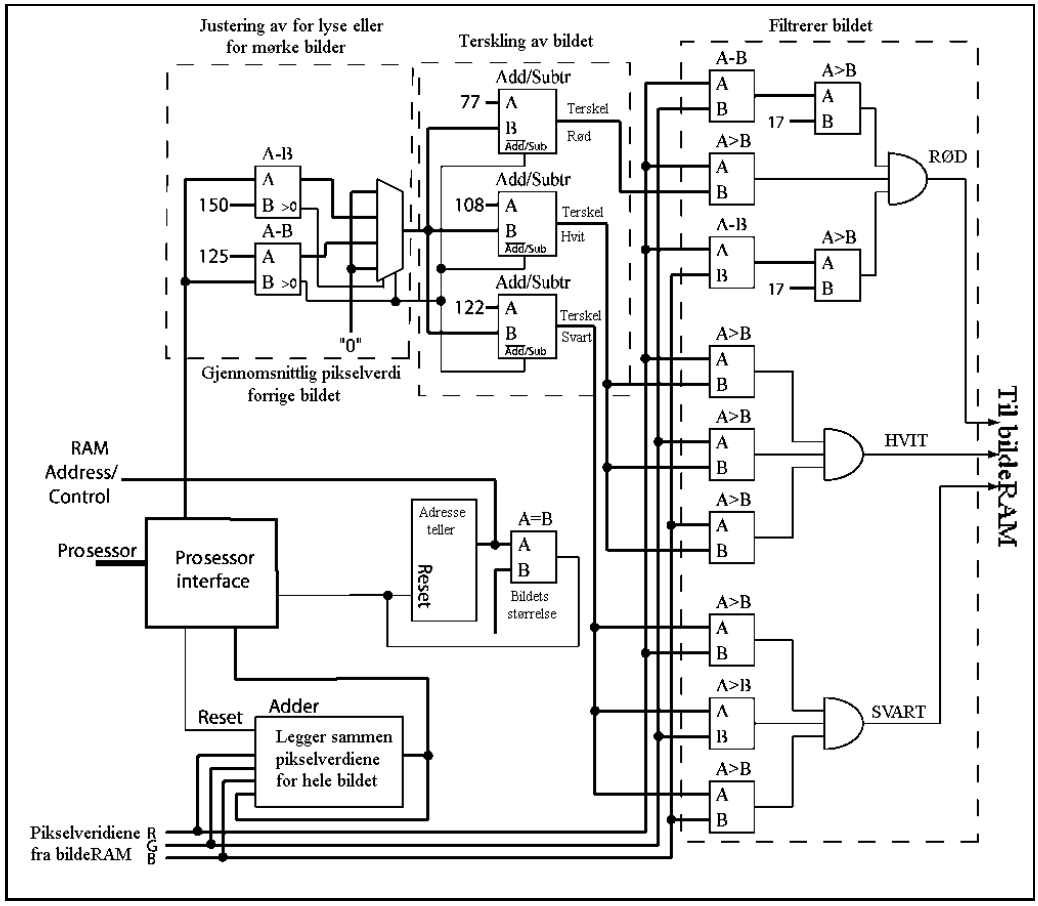
Figur 5.4: De filtrerte fargene

Input	MUX kontroll signal	MUX utgang	Kommentar
I_0	0 0	0	
I_1	0 1	125 - Gjennomsnittlig pikselverdi	For mørke bilder
I_2	1 0	Gjennomsnittlig pikselverdi - 150	For lyse bilder
I_3	1 1	0	

Tabell 5.1: Multiplekserens operasjoner

det skal deles på antall piksler i hele bildet (307200) for å finne gjennomsnittlig pikselverdi. Divisjon er en kompleks operasjon å utføre på en FPGA, men en prosessor er velegnet for en slik operasjon. I og med at en prosessor skal gjøre resten av gjenkjenningen utenfor FPGAen, kan utregningen gjøres i prosessoren, og resultatet sendes tilbake til FPGA-en.

Det blir sjekkes om gjennomsnittet av pikslene er for lyst eller mørkt, slik som beskrevet i avsnitt 4.3.2. Dersom gjennomsnittlig pikselverdi skal legges til grunn for lysutjevningen av *samme* bildet, får dette konsekvenser for tidsforbruket. Bilder som har gjennomsnittsverdi som er høyere eller lavere enn det akseptable intervallet, må fargefiltreres en gang til med de nye grenseverdiene. Det er derfor valgt å legge gjennomsnittsverdien for *forrige* bilde til grunn for lysutjevningen. Dette vil ha minimal betydning for gjenkjenningen. Intervallet mellom bildene skal som nevnt være under 1/10 sekund, og lysforholdene forandres ikke mye på denne tiden. Etter at grensene er oppdatert, begynner filtreringen av neste bilde med grenseverdiene som er beregnet ut fra forrige bilde. Hvordan multiplekseren (MUX), som brukes til å justere grenseverdiene i figur 5.5, opererer er beskrevet i tabell 5.1.



Figur 5.5: Blokkdiagram av fargefiltrering på FPGA [25]

5.6 Ytelse på FPGA

Resultatene fra ytelsestesten er hentet fra filer som er generert i Xilinx project navigator. VHDL koden ble syntetisert før det ble gjort "place and route" med hensyn på Xilinx Virtex II Pro (XC2VP7-FG456-7). Det viser seg at 3% av CLBene og 35% av de tilgjenglige IOBene benyttes for å realisere fargefiltrering på FPGA. Dette betyr at det er plass til flere faser av IDSL i denne FPGAen. I IDSL er det ønskelig å benytte en billig prosessor, som dermed er tregere enn den som hittil er benyttet [25]. For at IDSL fortsatt skal ha den samme hastigheten med en tregere prosessor, er det nødvendig at flere metoder gjøres på FPGAen.

Tiden det tar å gjennomføre selve fargefiltreringen er avgjørende for om målsetningen oppnås. Filer som er generert under "place and route" viser at FPGAen må ha klokkeperiode på minimum 7.5ns, som tilsvarer en klokkefrekvens på 133MHz. Fargefiltreringen benytter 6 klokkeperioder for å filtrere *ett* piksel. Siden bildet består av 307200 (640x480) piksler, vil filtrering av ett bilde med klokkeperiode på 8ns ta:

$8 \cdot 6 \cdot 307200 = 14\,745\,600$ ns (ca 15 ms)

Dette betyr at filtrering på FPGA bruker 32% av den tiden det tar for prosessoren å fargefiltrere. Dette betyr igjen at gjenkjenning av ett bilde tar ca 95 ms, som betyr at IDSL kan sjekke 10,5 bilder per sekund. Dette er innenfor målsettingen vi har satt med minimum 10 bilder per sekund.

5.6.1 Samlebånd

Samlebånd oppsett (eng. pipelining) er et meget effektivt oppsett for å øke hastighet. Samlebånd går ut på at en stor oppgave, deles opp i mindre deler som kan gjøres delvis samtidig i uavhengige enheter. For å gjøre dette klarere forklares samlebånd med et eksempel. Det skal vaskes, tørkes og strykes tøy. Hvis det gjøres uten samlebånd oppsett blir alle plaggene vasket, så blir de tørket og til slutt strøket. Neste jobb blir ikke påbegynt før forrige er avsluttet. Prosessen kan effektiviseres med samlebånd oppsett ved å sette i gang ny vask mens den første vasken er i tørketrommelen. Dette fører til at neste jobb kan påbegynnes samtidig som neste steg av første jobb starter. Man får parallelle prosesser og ikke etterfølgende, sekvensielle prosesser. Det er som samlebånd, der det blir jobbet med forskjellige deler på forskjellige steder.

Det er mulig for IDSL å benytte samlebånd oppsett. Dette gjøres ved at

prosessoren begynner å jobbe med et filtrert bilde, samtidig som FPGA-en begynner å filtrere neste bilde. I og med at prosessoren bruker lenger tid enn FPGAen på hvert bilde, er prosessoren flaskehalsen i systemet. Gjenkjenningstiden av et bilde vil gå ned, og være det samme som tiden prosessoren bruker på resten av gjenkjenningen. I avsnitt 4.7 er det beskrevet at denne tiden ca 80 ms. Det tilsvarer at 12,5 bilder sjekkes per sekund.

Kapittel 6

Videreutvikling av IDSL

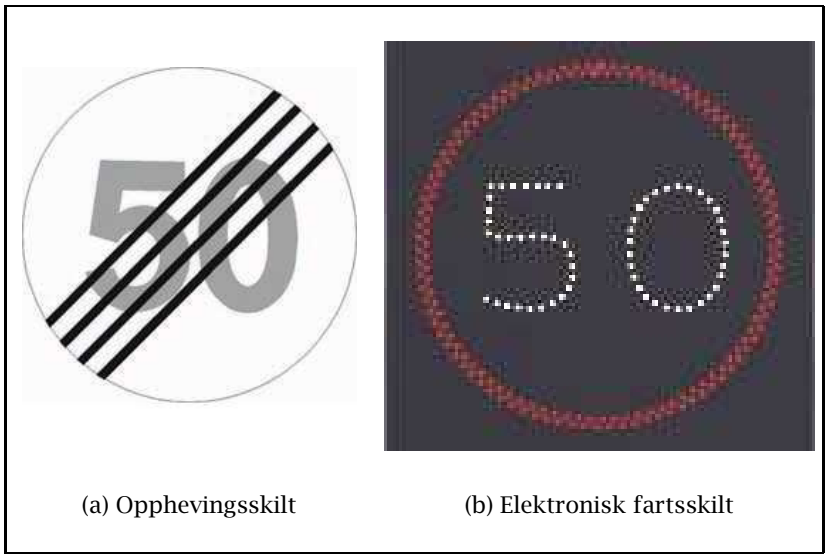
Før IDSL taes i bruk bør det foretas ytterligere forbedringer. I dette kapittelet blir disse forbedringene beskrevet.

6.1 Andre typer fartsskilt

Så langt er IDSL i stand til å gjenkjenne en type fartsskilt. Det finnes tre typer fartsskilt i Norge. Foruten den ordinære typen IDSL er utviklet for, finnes det opphevingsskilt og elektroniske fartsskilt, som vist i figur 6.1. Det er ikke laget metode for å gjenkjenne de sistnevnte.

6.1.1 Opphevingsskilt

Opphevingsskilt er en type skilt som brukes til å forteller førere at det er slutt på den nåværende fartsgrensen, og fartsgrensen blir satt til standardgrensen. I *Norge* er standardgrensene 50 og 80. Etter at opphevingsskiltet er passert, er fartsgrensen 50 dersom man er i tettbebyggelse, og 80 dersom man er på landevei. Opphevingsskiltet er hvitt, grått og svart. Gjeldene fartsgrense står i grått med fire diagonale svarte linjer over hele skiltet. Et opphevingskilt for 50 sone er vist i figur 6.1 (a). Det er klart at opphevingsskilt ikke blir funnet med metoden som er beskrevet i denne oppgaven. Dette fordi metoden benytter seg av rødfargen som finnes i fartsskiltet. For at opphevingsskilt skal være mulig å tyde, må enten den



(a) Opphevingsskilt (b) Elektronisk fartsskilt

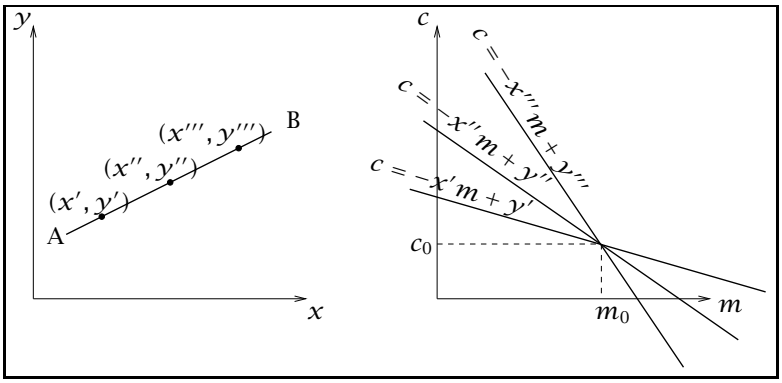
Figur 6.1: Opphevingsskilt og elektronisk fartsskilt

eksisterende metoden utvides eller det må lages en metode i tillegg som finner opphevingsskilt.

Det som er spesielt for opphevingsskilt er de tre diagonale svarte linjene. Jeg ser for meg at Hough-transformasjon kan være en mulig metode for å finne disse. Hough-transformasjonen må benyttes på en annen måte enn den som er forklart i avsnitt 2.3.1. I figur 6.2 er Hough-transformasjon illustrert. Hvert punkt på en linje i bildet overføres til et annet koordinatsystem, som har c og m på aksene. I dette koordinatsystemet beskrives et punkt i bildet med en linje. Dersom et punkt i bildet har koordinater i $x = 3$ og $y = 4$, beskrives dette punktet ved linjen $c = -3m + 4$ i koordinatsystemet. Punkter som ligger på en linje i bildet vil alltid krysses i ett og samme punkt i koordinatsystemet. Dermed vet man hvor linjen går. Dette kan benyttes for de fire diagonale linjene i opphevingsskiltet.

6.1.2 Elektriske fartsskilt

Elektriske fartsskilt benyttes ofte i tunneler og steder hvor det skal være mulighet for å forandre fartsgrensen raskt eller ofte. Disse skiltene har en annen fargeoppbygning enn vanlige fartsskilt. Den røde sirkelen rundt selve skiltet, er lik vanlige fartsskilt. Derimot er det hvite siffer på svart bunn inne i sirkelen. Et bilde av denne typen skilt er vist i figur 6.1 (b). Denne sammensetningen av farger fører til at metoden som er utvik-



Figur 6.2: Illustrasjon av Houg-transformasjon

let i denne oppgaven ikke vil gjenkjenne elektroniske fartsskilt. Et annet problem med de elektroniske fartsskiltene, er at det er lysdioder som viser tallene. Dette kan fører til at tallet ikke får et homogent områdét i bildet, noe som kan føre til at gjenkjenningen av skiltet vanskeliggjøres.

6.2 Sjekke begge tallene i skiltet

Slik gjenkjenningen av skilt foretas av IDSL, blir bare det første sifferet i skiltet brukt for å finne fartsgrensen. Det andre sifferet har ingen betydning for om riktig fartsgrense blir funnet eller ikke. Det siste sifferet kan brukes som en verifikasjon på at det faktisk er fartsskilt som er funnet. Det er slike tilfeller hvor det kan være nyttig å benytte det siste sifferet. Både plassering og størrelse på det siste sifferet er kjent, så samme metode som blir brukt for å gjenkjenne det første sifferet, kan benyttes til å gjenkjenne det andre.

De bildene som *ikke* inneholder skilt, men hvor IDSL oppfatter at det finnes skilt, vil bli eliminert ved å teste at det siste sifferet er null. Dette vil føre til at korrekt klassifikasjon av testbildene er på 91,9% [26] For at dette resultatet skal oppnåes er det ikke tatt hensyn til fartsskilt som kan refuseres som fartsskilt ved dårlig ekstrahert '0'.

6.3 Flere templater

Under testing av IDSL viste det seg at templatene som er valgt fant de aller fleste fartsskiltene. Som nevnt brukes 6 templater med en viss forskjell i størrelse. Det viser seg at flere av fartsskiltene i testbildene har størrelse som havnet mellom to templater. Den eneste grunnen til at disse fartsskiltene blir funnet, er fordi kravet for at objektet stemmer med templat ikke er spesielt høyt. Dette fører også til at enkelte objekter som ikke er skilt kommer ut av templatsjekken som skilt. Disse objektene blir stort sett avkreftet som skilt av tester, men enkelte kommer også forbi disse. Det er mulig flere av de "falske" fartsskiltene blir eliminert dersom kravet for sammenlikning mellom templat og objekt blir strengere. Den eneste måten å finne alle skiltene og samtidig ha strengere krav, er å sjekke objektene med flere templater. Sjekk av templater har vist seg ikke å være spesielt tidskrevende, men den lureste løsningen vil være å implementere templatsjekken på FPGA. I følge [21] kan templatsjekk gjøres meget effektivt på FPGA.

6.4 Kriterier for å bytte grense

Som nevnt i kapittel 1 er det mulig å fastsette flere kvalitetskrav til hva IDSL skal finne før fartsgrensen endres. Slike krav vil føre til at gjenkjenningen blir mer pålitelig og sjansen for å feilinformere føreren går ned.

6.4.1 Verifikasjon

Før det blir konkludert med bytte av fartsgrense, bør fartsgrensen verifiseres. IDSL kommer til å sjekke over 10 bilder i sekundet, og det kan derfor stilles krav til at fartsgrense bli funnet i flere etterfølgende bilder. Kravet kan være at det skal bli funnet samme fartsgrense i fem av syv etterfølgende bilder. Dersom det behandles 10 bilder i sekundet, betyr det at bilen har kjørt 9,7 meter mens IDSL har sjekket syv bilder, hvis bilen kjører i 50 km/t. Dette betyr at systemet kan feile flere ganger på samme skilt før bilen passerer, og allikevel blir det funnet fartsgrense i fem av syv etterfølgende bilder.

6.4.2 Hensyn til skiltbestemmelser

I Norge finnes det regler for hvilke fartsgrenser som kan etterfølge hverandre. Det vil si at 80 sone aldri etterfølges av 30 sone. Dersom IDSL tar hensyn til disse reglene, vil det innskrenke mulige skilt som til enhver tid kan forekomme. Dette kan være med på å forhindre at fartsgrense settes feilaktig. Dette kan også være med på å forhindre at skilt som hører til veier som går parallelt vil bli funnet.

6.4.3 Skilt på begge sider

Det kan i enkelte situasjoner være nyttig at det er fartsskilt på begge sider av veien når fartsgrensen forandres på veistrekningen man kjører på. Store kjøretøyer kan ligge mellom bilen og et skilt, slik at føreren ikke får med seg at det er skilt i veikanten. Det er også mulig for IDSL å benytte seg av denne regelen. I IDSL kan det være krav om at fartsgrensen ikke byttets uten at det er funnet skilt med samme fartsgrense på begge sider av bilen. Dette vil føre til at skilt som hører til veier som går parallelt ikke tolkes til å høre til veistrekningen bilen kjører på. Dersom det blir funnet fartsskilt i høyre halvdel av et bildet, sjekkes bare venstre halvdel av neste bildet, om det inneholder fartsskilt. Ved å kombinere dette med forskrifter om lovlige skilt, som er beskrevet i avsnitt 6.4.2, vil gjenkjenningen av fartsskilt bli sikrere. I en enkelte situasjoner endres fartsgrensen uten at det er fartsskilt på begge sider av veien. Dette gjelder når man kjører på vei, slik som motorvei. I slike tilfeller vil det bare være skilt på den ene siden av veien. IDSL må derfor være i stand til å skifte fartsgrense selv om det bare blir funnet fartsskilt på den ene siden av veien.

6.5 Ulike minneteknologier

FPGA-oppsettet som er vist i figur 5.3 kan gjøres mer effektivt. Under beskrives forskjellige typer minneteknologier som kan forbedre ytelsen ytterligere. Selve fargefiltreringen tar ikke lang tid, derfor kan ytelsen forbedres ved å benytte RAM som kan overføre større mengder data i løpet av en gitt tidsperiode.

6.5.1 DDR RAM

Double Data Rate (DDR) RAM er en forholdsvis ny teknologi som gjør det mulig å overføre data vesentlig raskere enn med vanlig RAM. Som navnet sier, er det mulig å overføre dobbelt så mye data med DDR RAM i forhold til vanlig RAM. Dette betyr at i en gitt tidsperiode kan DDR RAM overføre dobbelt så mye informasjon som annen RAM.

6.5.2 Burst

Burst er en teknologi som kan forbedre hastigheten på overføringen mellom FPGA og RAM. Burst er noe som de fleste av dagens RAM har mulighet til. Teknologien går ut på at man ikke bare sender innholdet i en lokasjon i RAM av gangen, men innholdet i en rekke lokasjoner sendes i rask rekkefølge. Lengden på disse rekkene varierer for de forskjellige RAMene, hvor de fleste kan sende rekker på 2 til 8 lokasjoner. Det som skjer er at man ber om innholdet i en lokasjon i RAM, og samtidig hvor mange av de etterfølgende lokasjonene som ønskes. I praksis betyr dette at datamengden som kan overføres øker betraktelig.

For RAM'en jeg har valgt å bruke, er det mulig å sende rekker på 1,2,4 eller 8 lokasjoner. I praksis vil det si at man gir beskjed til RAM om at den skal sende det som ligger i lokasjon fire, og de fire neste. RAM bruker litt tid på å finne frem til den første og sende den, så kommer de neste tre verdiene i rask rekkefølge.

Slik oppsettet virker i dag, brukes det seks klokkeperioder fra det blir bedt om innholdet i en lokasjon, til den er inne i FPGAen. Dette betyr at det tar 24 klokkeperioder for å sende fire lokasjoner. Dersom burst benyttes, vil denne tiden være redusert til ni klokkeperioder, seks på den første, og en for hver av de tre neste. Det samme gjelder når svaret skal skrives til RAM. Man sier hvilken lokasjon svaret skal til, og hvor mange svar som kommer, så vil RAM'en selv lagre svaret i den angitte lokasjonen, og de tre neste.

6.5.3 Lagre resultatet internt på FPGA

De bildene som brukes i testingen har en størrelse på 640x480 piksler. I og med at resultatet etter filtrering av ett piksel krever tre bit, vil resultatet etter fargefiltrering kreve 116 kbyte for hele bildet. Dette er

ikke veldig mye og det finnes FPGA'er som har intern lagringsplass på denne størrelsen. Dersom videre arbeid skal gjøres med resultatene inne i FPGAen, vil denne behandlingen gå mye raskere enn om resultatet skal hentes fra ekstern RAM. Selve fargefiltreringen vil ikke bli påvirket av denne forandringen. Dette vil bare være hensiktsmessig dersom det er mer enn fargefiltrering som skal gjøres på FPGA'en. Dersom eksterne komponenter skal bearbeide resultatene videre er det en fordel å sende resultatet til ekstern RAM. Dermed kan eksterne medier få direkte tilgang til resultatet.

6.6 Videokamera

Når det skal monteres kamera i bilen, er det viktig at det er justert slik at det er sikkert at skilt blir filmet. Enkelte systemer har hatt problemer med at bilder som blir tatt med fastmontert kamera i bil ikke er gode nok [6, 14]. Grunnen til det er at kjøretøyet beveger seg, noe som fører til at kameraet beveges. Dermed blir bildene uskarpe, som vanskeliggjør gjenkjenning av eventuelle skilt. Derfor kan det være nødvendig å finne en stabilisator som kameraet kan kobles til får å motvirke denne effekten. Dette byr på store utfordringer, siden det er veldig mye bevegelse i selve bilen. Det finnes kameraer som absorberer vibrasjoner når det holdes, men det er ikke tilstrekkelig når kameraet er montert i en bil og skal kunne ta bilder mens bilen beveger seg på veien.

6.6.1 Dual Port RAM

Dual Port RAM, er en annen type RAM det er mulig å bruke. Forskjellen mellom denne typen RAM og de som er beskrevet tidligere, er at det er mulig å både skrive og lese til RAM på samme tid.

Det kan være hensiktsmessig å benytte Dual Port RAM. Grunnen til det er at når kameraet leverer bildene som skal sjekkes, vil det være ønskelig at dette ikke påvirker resten av systemet. Ved å koble kameraet rett på dual port RAM, vil kameraet kunne skrive verdiene til neste bilde inn i RAM, samtidig som forrige bilde blir hentet av FPGA'en. Ved denne type oppsett er det en fordel å benytte en RAM til bilde, og den andre til resultat.

Det vil ikke bare være mellom kamera og FPGAen det vil være en fordel med dual port RAM. Det vil også være en fordel med slik type RAM

i grensesnittet mellom FPGA og prosessor. Alle enhetene i IDSL vil på denne måten kunne arbeide nesten uten å ta hensyn til andre enheter.

Kapittel 7

Konklusjon

I denne oppgaven har det blitt beskrevet et system som kan gjenkjenne norske fartsskilt. Det er beskrevet hvordan et system kalt IDSL (Image Detection of Speed Limits), som er utviklet av Lukas Sekanina i samarbeid med Jim Tørresen, finner fartsskilt ut fra stillbilder.

IDSL er videreutviklet for å få det robust og raskt. For å få korrekt klassifisering av 90% av bildene er det gjort forandringer i det opprinnelige programmet. Det blir beskrevet hvilke forandringer som er gjort med de forskjellige metodene i gjenkjenningsprosessen, og hvilken påvrikning disse har for gjenkjenningsprosessen. Tiden programmet bruker i gjennomsnitt, for å sjekke bilder som inneholder fartsskilt er 121 ms. For bilder som ikke inneholder fartsskilt er gjennomsnittligtidsforbruk 123,7 ms.

Et av formålene med denne oppgaven har vært å gjøre IDSL klar for prototyptesting. Det er stilt krav om at minst 10 bilder skal sjekkes per sekund. Dette er ikke oppnådd ved å gjøre hele gjenkjenningsprosessen på en datamaskin. Det er derfor kartlagt hvilke faser av programmet som bruker mest tid. Det viste seg at fargefiltrering er den fasen som bruker lengst tid. Fargefiltrering er derfor testet på en Field Programmable Gate Array (FPGA) for å se om dette reduserer tidsforbruket tilstrekkelig.

Det viste seg at metoden som filtrerer bildet bruker ca 50 ms på hvert bilde når den gjøres av prosessor. Resultatene fra tester på FPGA viser at fargefiltreringen vil bruke 15 ms på å filtrere et bilde. Dermed bruker IDSL ca 95 ms på å gjenkjenne ett bilde. Dette vil si at IDSL kan sjekke ca 10,5 bilder per sekund. Dette kan ytterligere forbedres ved å benytte samlebånd oppsett. Gjenkjenning av et bilde kan gjøres av prosessoren,

samtidig som neste bildet filtreres av FPGAen. Med dette oppsettet vil det være mulig å sjekke 12,5 bilder per sekund.

Tillegg A

VHDL Code

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use IEEE.STD_LOGIC_ARITH.ALL;

entity farg_filt is

--Data kommer inn på bussen med 9 bit til hver farge, og er på formen:
--RRRRRRRRRGGGGGGGGGBBBBBBBBB.
--Dette fører til at 26 downto 18 er røde
--17 downto 9 er grønne
--8 downto 0 er blå
--hvit og svar 110
--ingen farger 000
--rød og svart 101
--rød          001
--svart        100
--hvit         010

-- integer datatype range is -2147483647 to +2147483647
-- som vil maksimal verdi for alle piksler er 78336000

  port (
    r_set:in    std_logic;                      -- Reset
    clk : in    std_logic;                      -- Ekstern klokke
    da  : inout std_logic_vector(26 downto 0);  -- Tristate databuss
    adr : out   std_logic_vector(18 downto 0);  -- Adressen til ram
    nCE : out   std_logic := '1';               -- low-active Chip-Enable of the SRAM device; def
    nOE : out   std_logic := '1';               -- low-active Output-Enable of the SRAM device; d
    CE2 : out   std_logic := '1';
    DATA: out   std_logic_vector(2 downto 0);
    g_snitt : in std_logic_vector(7 downto 0); --gjenomsnittet delingen gjøres utenfor FPGA'en
    sumout: out std_logic_vector (28 downto 0));

end farg_filt;
```

```
architecture behavior of farg_filt is

  signal ADR_COUNT : std_logic_vector(18 downto 0) := (others => '0');  --Teller for å f
  signal INC_ADR   : std_logic := '0';  -- Bestemmer om adressen skal økes
  signal LES       : boolean   := FALSE; -- Om det skal leses fra ram
  signal OE        : std_logic := '0';  -- Om det skal skrives ut på bussen, aktiv høy
  signal tmp       : std_logic := '0';  --er inv_clk, som trengs til lesing fra ram
  signal DATA_IN   : std_logic_vector(26 downto 0) := (others => '0');  --Data
inn fra bussen
  signal val_dat   : std_logic := '0';   -- Bestemmer om det er klart for fargefiltrerir
  signal DIFF      : integer range -125 to 105 ;  -- Lysutgjevning
  signal sum    : integer range 0 to  235008000 ;  -- sum av pikselverdiene
  signal rod    : integer range 0 to 255;
  signal bla    : integer range 0 to 255;
  signal gronn  : integer range 0 to 255;
  signal g_rod   : integer range 0 to 255;
  signal g_svart : integer range 0 to 255;
  signal g_hvit  : integer range 0 to 255;
  signal snitt   : integer range 0 to 255;

  type state_type is (IDLE, READ1, READ2, READ3, READ4);
  signal STATE: state_type;

begin  -- behavior

  adr <= ADR_Count;

tilst_styring: process (clk, r_set)
begin  -- process ram_controller
  if r_set = '0' then                    -- asynchronous reset (active low)
    LES <= true;
    state <= idle;
  elsif clk'event and clk = '1' then    -- rising clock edge

     g_rod   <= 77  + diff;
     g_svart <= 122 + diff;
     g_hvit  <= 108 + diff;
    INC_ADR <= '0';
    OE <= '0';
    val_dat <= '0';
    les <= false;

     case STATE is
      when IDLE =>
        if (LES = true) then
          STATE <= READ1;
        else
          STATE <= IDLE;
        end if;

      when READ1 =>
        STATE <= READ2;
```

```
      when READ2 =>
        STATE <= READ3;

      when READ3 =>
   data_in <= da;
        STATE <= READ4;

       when READ4 =>  --venter en periode for å være sikker på at data er inne
        val_dat <= '1';
   rod   <= conv_integer (DATA_IN(26 DOWNTO 18));
        gronn <= conv_integer (data_in(17 downto 9));
        bla   <= conv_integer (data_in(8 downto 0));
        les <=true;
        INC_ADR <= '1';
        STATE <= IDLE;

      when others =>
        STATE <= IDLE;

    end case;
  end if;
end process tilst_styring;

ram_styring:
process (STATE, tmp)  --styrer alle styresignaler til ram
begin
          if (state = idle) then
            CE2 <= '1';
            nCE <= '1';
            nOE <= '1';
          elsif (state = READ1) then --and tmp = '0') then
            CE2 <= '1';
            nCE <= '0';
            nOE <= '1';
          elsif (state = READ2) then  --read1 and tmp = '1') then
            CE2 <= '1';
            nCE <= '0';
            nOE <= '0';
          elsif (state = READ3) then
            CE2 <= '1';
            nCE <= '0';
            nOE <= '0';
          else
            CE2 <= '1';
            nCE <= '1';
            nOE <= '1';
          end if;
end process;

FILTERING:      --Her gjøres fargefiltreringen
  process (val_dat, r_set)

  begin
  if r_set = '0' then
```

```
  sum <= 0;
  else
    if val_dat'event and val_dat = '1' then
      DATA <= (others => '0');

   sum<=sum+rod+gronn+bla;

      --RØD
      if (rod > g_rod) and (rod - gronn > 17) and (rod -bla > 17) then
        DATA(0 downto 0) <= "1";
      end if;

      --svart
      if (rod < g_svart and gronn < g_svart and bla < g_svart ) then
       DATA (2 downto 2) <= "1";
      end if;

       --hvit
      if (rod > g_hvit and gronn > g_hvit and bla > g_hvit ) then
        DATA (1 downto 1) <= "1";
      end if;
    end if;
end if;
  end process;

ADDRESS_COUNTER :
  process (CLK,r_set,INC_ADR)

  begin
    if r_set ='0' then
      DIFF <= 0;
 snitt <= 0;
      ADR_COUNT <= (others => '0');
    elsif rising_edge(CLK) then
      if (INC_ADR = '1') then
        if (ADR_COUNT = "1001011000000000000") then --Bildets størrelse
    snitt <= conv_integer (g_snitt);
sumout <= CONV_STD_LOGIC_VECTOR(sum, 29);
          if (snitt < 125) then
            diff <= snitt-125;
          elsif (snitt > 150) then
            diff <= snitt - 150;
          end if;
          ADR_COUNT <= (others => '0');
        else
          ADR_COUNT <= ADR_COUNT + 1;
        end if;
      end if;
    end if;
  end process;

    -- Tristate control
da <= (others => 'Z');
```

```
end behavior;
```

Tillegg B

Informasjon om CD

```
CDen inneholder:
----------------

På CDen er det fire mapper: program, template, test_m_fskilt
og test_u_fskilt.

Program mappen:

-IDSL.exe er en kjørbar versjon av IDSL. Denne filen kan
 kjøres lokalt på maskinen etter at enkelte lokale
 forberedelser, som er forklart under, er gjort.


-IDSL.exe er selve IDSL. Denne filen kan kjøres lokalt på
 maskinen etter at enkelte lokale forberedelser, som er
 forklart under, er gjort.

-ViewFrm.cpp er hovedkoden til IDSL. Det er her koden til all
 behandling av bilder ligger.

-ssl.bpr er toppfilen til prosjekt. Dette er filen som samler
 hele prosjektet hierarkisk. Denne filen kan bare åpnes med
 Borland C++ Builder.

Template mappen:

Denne mappen inneholder de 6 templatene som benyttes for å
lokalisere den røde ringen i bildet.
```

Test_m_fskilt mappen:

Denne mappen inneholder bilder som inneholder fartsskilt.
Disse bildene er benyttet under utviklingen og testing IDSL.

Test_u_fskilt mappen:

Denne mappen inneholder bilder som ikke inneholder fartsskilt.
Disse bildene er benyttet under utvikling og testing av IDSL.

Hvordan få IDSL til å virke på maskinen:

For at IDSL skal virke på maskinen du sitter på, er det behov for å gjør noen enkle ting. Det må lages to mapper:

- c:\IDSL\res
- c:\IDSL\template

Etter at disse to mappene er laget, må alle filene som ligger i template mappen på CDen kopieres over til c:\IDSL\template. Det må også lages en fil. Denne filen MÅ hete res.txt. Denne filen skal plasseres i c:\IDSL\res\. Nå dette er gjort vil IDSL virke. Informasjon om gjenkjenningsprosessen blir lagret i c:\IDSL\res\res.txt

Etter at IDSL er startet, kan man velge bilde man ønsker å sjekke. Hvis det er ønskelig å sjekke flere enn ett, må flere markeres. Denne markeringen må skje ved at det nederste bilde markeres først. Sekvensen av bilder må markers oppover uten å ‘‘hoppe over’’ bilder.

Bibliografi

[1] Fritz Albregtsen. http://www.ifi.uio.no/in106/forelesninger/fa_2.ps, februar 2004.

[2] Oliver Carsten og Fergus Tate. Intelligent speed adaption: The best collision avoidance system? *17 th International technical conference on the enhanced safety of vehicles*, side Paper 324, 4-7 juni 2001.

[3] Bruce A. Draper, J. Ross Beveridge, A. P. Willem Böhmand Charles Ross og Monica Chawathe. Accelerated image processing on FPGAs. *IEEE transactions on image processing*, 12(12):1543–1551, Desember 2003.

[4] Yinzi Du, Chein l Chang og Paul David Thouin. Unsupervised approach to color video thresholding. *Socity of Photo-Optical Instrumentation Engeneers*, 2(43):282–289, February 2004.

[5] E.Jamro og K. Wiatr. Convolution operation inplemented in FPGA structures for real-time image processing. *Proc. of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA'2001)*, side 417–422, 2001.

[6] C. Y. Fang, C. S. Fuh, S. W. Chen og P. S. Yen. A road sign recognition system based on dynamic visual model. *Computer vision and pattern recognition*, 2003.

[7] Luke Fletcher, Nicholas Apostoloff, Lars Petersson og Alexander Zelinsky. Vision in and out of Vehicles. *IEEE Intelligent Systems*, 18(3):12–17, June 2003.

[8] Dr. Edward A. Fox. Color and image processing. http://courses.cs.vt.edu/~cs4624/s98/sspace/imgproc/, 15 april 2004.

[9] Rafael C. Ganzalez og Richard E. Woods. *Digital Image Processing*, kapittel 7. Addison-Wesly Publishing Company. Inc, 1992.

[10] May Linn Gerding. Bilen tar styringen. *VG*, 30. November, 2003.

[11] Willie D. Jones. Keeping cars from crashing. *IEEE spectrum*, September, 2001.

[12] Logic Systems Laboratory. http://lslwww.epfl.ch/~aperez/FAST/slide20.html, 23 mars 2004.

[13] Marc Lalonde og Ying Li. Road sign recognitinon, survey of the state of the art. *CRIM/IIT (Centre de recherche informatique de Montreal)*, 1995.

[14] Han Liu, Ding Liu og Jing Xin. Real-time recognition of road traffic sign in motion image based on genetic algorithm. *Machine Learning and Cybernetics*, 1:83–86, 4-5 Nov. 2002.

[15] David Marshall. Region growing. http://www.cs.cf.ac.uk/Dave/Vision_lecture/node35.html, 25 april 2004.

[16] Stephanie McBader og Peter Lee, redaktører. *An FPGA Implementation of a Flexible, Parallel Image Processing Architecture Sutable for Embedded Vision Systems*, 2003.

[17] Jun Miura, Tsuyoshi Kanda og Yoshiaki Shirai. An active vision system for real-time traffic sign recognition. *Intelligent Transportation Systems*, 1-3 okt 2000.

[18] Masashi Nakamura, Syusaku Kodama, Takashi Jimbo og Masayoshi Umeno. Searching and recognition og road signpost using ring detection network. *IEEE SMC '99 Conference Proceedings*, side 190–195, 12-15 Oct. 1999.

[19] Hirofumi Ohara, Ikuko Nishikawa, Shigeto Miki og Noboru Yabuki. Detection and recognition of road signs using simple layerd neural networks. *9th international conference on neural information processing, Vol. 2*, side 626 – 630, 18-22 nov 2002.

[20] Lluis Pacheco, Joan Batlle og Xevi Cufi. A new approach to real time traffic sign recognition based on colour information. *Intelligent vehicles symposium, Paris*, side 339–344, 1994.

[21] Lukas Sekanina og Jim Tørresen. Detection of Norwegian speed limit signs. *Proc. of the 16th European Simulation Multiconferance*, Juni 2002.

[22] Statistisk Sentralbyrå. Statistisk årbok 2003. Rapport, Statistisk Sentralbyrå, 2003.

[23] Milan Sonka, Vallav Hlavac og Roger Boyle. *Image Processing, analysing and machine vision*. Cole publishing company, 1999.

[24] Rik Thomas. Less is more. *IEEE Review*, 49(5), May 2003.

[25] Jim Tørresen, Jørgen W. Bakke og Lukas Sekanina. Efficient image filtering and information reduction in reconfigurable logic. Submitted to conference, 2004.

[26] Jim Tørresen, Jørgen W. Bakke og Lukas Sekanina. Efficient recognition of speed limit signs. Submitted to conference, 2004.

[27] Jim Tørresen, Jørgen W. Bakke og Lukas Sekanina. Recognizing speed limit sign numbers by evolvable hardware. Submitted to conference, 2004.

[28] S. Vitabile, G. Pollaccia, G. Pilato og F. Solbello. Road signs recognition using a dynamic pixel aggregation tecnhique in the HSV color space. *11th International Conference*, side 572–577, 28 Sept 2001.

[29] Bilde av Xilinx Spartran-IIE. http://www.ednjapan.com/news/200111/20011126xillinx_spartanIIE.html, 5 april 2004.

[30] Xilinx (fpga produsent). http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Virtex-II+Pro+FPGAs, 5 april 2004.

[31] Vhdl sram models. http://tech-www.informatik.uni-hamburg.de/vhdl/models/sram/sram.html, 12 november 2003. Laget av Andre Klindworth på Universitetet i Hamburg.

[32] 3d plotet. http://frsl06.physik.uni-freiburg.de/privat/stille/kpl/kpldemo.html, 22 April 2004.

[33] General digitization information. http://www.ohiomemory.org/om/digimaging.html, 10 april 2004.