Irisgjenkjenning under varierende forhold

Hans Christian Sagbakken

Masteroppgave
Master i medieteknikk
30 ECTS
Avdeling for informatikk og medieteknikk
Høgskolen i Gjøvik, 2007

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Faculty of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Biometric systems refer to technologies that measure and analyze human physical characteristics. The most widely used characteristics are extracted from fingerprints, irises and retinas, facial patterns and hand measurements. Iris recognition is regarded as the most reliable biometric of these characteristics. Nowadays, most of the commercial iris-based identification systems use algorithms developed by Daugman. The Daugman advertised recognition rate are excellent. They are however, very likely measure under ideal conditions. Our main goal in this work is to test Daugman and other filtering algorithms proposed in the literature under varying conditions and compare their performances.

This document describes the work done for the master thesis during the spring 2007. The thesis focuses on iris-based identification under various conditions. The aim of this project is to examine under which conditions iris recognition is possible, and which filtering algorithm performs best under each unfavorable condition.

The iris recognition process usually consists of four major steps. The first step is to segment the iris out of the image containing the eye and part of the face, which localizes the iris pattern. Step two is the normalization, here the iris pattern will be extracted and scaled to a predefined size. Step tree is the encoding phase, here the details of the iris are filtered, extracted and represented in an iris code. The last step is the comparison, where two iris codes will be compared and a similarity score is computed. In this thesis we have focused on the encoding and filter algorithms, in step tree, under different unfavorable conditions. We used the open source code of Libor Masek and extend it with different filtering algorithms. The filters which were included in the analysis are: two Haar filter, one Log-Gabor filter and one Laplacian of Gaussian filter; which generate iris codes with sizes 702 bit, 87 bit, 9600 bit and 9600 bit respectively. The filtering algorithms have been tested using a database of 500 iris images. The images in the database have been corrupted using different degradation models. We used four different degradation models: additive Gaussian noise and blur, changing the light intensity and rotating the images. Two performance measures were used: the False Acceptance Rate (FAR) and False Rejection Rate (FRR). The first is estimated using inter-class comparisons while the second is estimated using intra-class comparisons. The total number of comparisons performed in the experiments are approximately seven million comparisons.

Based on the experimental results we obtained in this work, we can conclude that the performance of all the tested algorithms is dramatically affected by the degradations. The major cause of the performance drop is the sensitivity of the segmentation process to such degradations. These degradations introduce segmentation errors in many of the images. The experimental results also show that the Log-Gabor and Laplacian of Gaussian filters are best under optimal conditions. Under non-optimal conditions however, the Haar filter with 702 bit representation achieves close to best result under most of the degradation

models. This maybe due to the fact that it extracts the iris characteristics based on fewer but most prominent details in the iris which are relatively more robust to degradations.

Sammendrag

Biometriske systemer er definert som teknologier som måler og analysere menneskets fysiske kjennetegn. De mest brukte kjennetegnene er hentet fra fingeravtrykk, retina og iris, ansiktsmønster og hånd målinger. Irisgjenkjenning er ansett som den mest pålitelige av disse biometriske teknologiene. Nå til dags bruker de mest komersielle iris identifikasjonsystemene algoritmer utviklet av Daugman. Disse algoritmene gir perfekt gjenkjenning, men er bare testet under optimale forhold. Vårt mål i dette arbeidet er å teste Daugman's og andre filteralgoritmer som er foreslått i litteraturen under varierende forhold og sammenligne deres ytelse.

Denne rapporten er skrevet for masteroppgaven som er gjennomført våren 2007. Oppgaven tar for seg irisgjenkjenning under varierende forhold. Målet med oppgaven er å undersøke under hvilke forhold det er mulig med irisgjenkjenning, og hvem filteralgoritme som egner seg best under ugunstige forhold.

Irisgjennkjennings prosessen består hovedsaklig av fire steg. Det første steget er å sementere irisen ut av bildet som består av øyet og en del av ansiktet, her lokaliseres irismønsteret. Steg to er normaliseringen, her blir irismønsteret hentet ut og skalert til en forhåndsdefinert størrelse. Steg tre er kodingsfasen, her blir detaljene i irisen filtrert, hentet ut og representert i en iriskode. Det siste steget er sammenligningen, her blir to iriskoder sammenlignet og en likhetsverdi blir beregnet. I denne oppgaven har vi fokusert på koding og filteralgoritmene, i steg tre, under forskjellige ugunstige forhold. Vi har brukt opensource koden til Libor Masek og utvidet denne med forskjellige filteralgoritmer. Filteralgoritmene som inngår i analysen er: to Haarfilter som genererer iriskoder på 702 og 87-bit, et Log-Gabor filter og et Laplacian of Gaussian filter som genererer iriskoder på 9600 bit. Filteralgoritmene har blitt testet mot en database bestående av 500 irisbilder. Bildene i databasen har blitt simulert ved forskjellige nedbrytingsmodeller. Vi har brukt fire forskjellige nedbrytingsmodeller: Lagt til Gaussian støy og blur, forandret lysintensiteten og rotert bildene. Det har blitt gjort to forskjellige målinger: False Acceptance Rate (FAR) og False Rejection Rate (FRR). Den første blir beregnet ved å gjøre interklasse-sammenligninger, mens den andre blir beregnet ved å gjøre intraklasse-sammenligninger. Totalt i eksperimentet har det blitt utført nærmere sju millioner sammenligninger.

Basert på eksperimentresultatene som er oppnådd i dette arbeidet, kan vi konkludere med at ytelsen til alle algoritmene som er testet blir veldig berørt av nedbrytingene. Hovedårsaken til dette er sensitiviteten i segmenteringen til slike nedbrytinger, som førte til segmenteringfeil i mange av bildene. Eksperimentresultatene viste også at Log-Gabor og Laplacian of Gaussian filterene er best under optimale forhold. Under ikke optimale forhold oppnådde vi totalt best resultat med 702-bit Haarfilter under de fleste nedbrytingsmodellene. Dette kan skyldes at 702-bit Haarfilteret henter ut detaljer fra irisen

basert på færre men mer synlige og fremstående detaljer, noe som gjør dette filteret mer robust under slike forhold.

Forord

Denne masteroppgaven er utført ved Institutt for informatikk og medieteknikk ved Høgskolen i Gjøvik. Jeg vil gjerne få takke mine veiledere ved Høgskolen i Gjøvik, Faouzi Alaya Cheikh og Jon Yngve Hardeberg for veldig god hjelp under arbeidet. Jeg vil også takke Ivar Farup for alle tips og råd i løpet av prosjektperioden.

Store deler av arbeidet i dette prosjektet har for min del omhandlet læring og tilegning av kunnskap. Kunnskap og innsikt i teorien og relaterte arbeider har vært en sentral del i prosjektet. Implementasjonen av forskjellige algoritmer for å utføre eksperimentene har også vært viktig og veldig lærerikt for meg.

Hans Christian Sagbakken, 2007/05/30

Innhold

Figurer

Tabeller

1 Introduksjon

Dette kapitlet gir en introduksjon til masteroppgaven. Kapitlet inneholder en innledning til problemområdet, motivasjon for oppgaven, forskningsspørsmålene som ønskes besvart og oppgavens avgrensning. Til slutt gir kapitelet en oversikt over hvordan rapporten er strukturert.

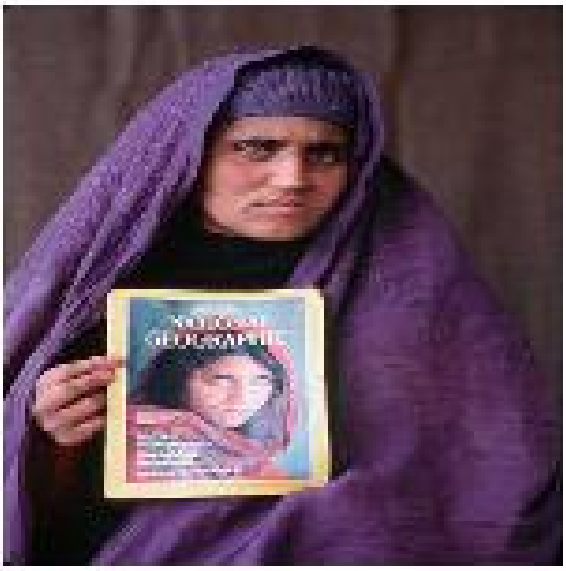
1.1 Innledning

Biometri er definert som måling av livsfenomener hos mennesker og dyr. Biometri brukes primært til autentisering, det vil si for å slå fast at en person er den hun hevder å være. Dette kalles ofte en-til-en matching, da man skal sammenligne den målingen brukeren avgir kun med den lagrede målingen til den hun hevder å være. Biometri kan også brukes til identifikasjon, det vil si å prøve å finne ut hvem en ukjent person er. Dette er kjent som en-til-mange matching, da målingen av personens biometriske egenskap må sammenlignes med tilsvarende fra alle personer som er registrert i en database. Hvis vedkommende finnes i basen vil man kunne få en match som avslører hvem den ukjente er.

I dagens samfunn bruker en passord og pinkoder for å få tilgang til et stort antall tjenester og områder. Et problem med dette er at mennesket har begrensede evner til å huske passord. Dette medfører ofte til at samme passord brukes flere steder, noe sikkerhetseksperter anser som usikkert. Biometrisk gjenkjenning har den fine egenskapen at brukeren ikke trenger å huske noe for å kunne verifiseres eller gjenkjennes. Det eksisterer i dag flere typer biometriske metoder basert på fingeravtrykk, håndtrykk og ansikts mønster. Felles for disse metodene er at livsfenomenene forandrer seg over tid og er ikke statiske hele livet.

Irisgjenkjenning er basert på avlesning av en persons regnbuehinne (iris), altså den fargede delen av øyet som omgir pupillen. Irisgjenkjenning er enkelt og effektivt i bruk og gir få falske matcher. Bruk av fargede eller bifokale kontaktlinser samt sterke briller kan være problematisk for slike systemer, og personer med enkelte øyesykdommer kan oppleve å ikke bli gjenkjent ved bruk av slik teknologi. Irisgjenkjenning er den biometriske teknologien som per dags dato er mest sikker for verifikasjon og identifikasjon av mennesker. Iris mønsteret utvikler seg i en periode fra fødselen og ett år frem i tid, og er etter denne perioden statisk resten av livet.

En kjent historie med irisgjenkjenning er identifikasjonen av en kvinnelig afgansk flykning som har preget forsiden av National Geographic magasinet siden 1985. Figur 1 viser et bilde av flyktningen.



Figur 1: National Geographic flykningen [1]

Hun ble fotografert av en reporter som jobbet for National Geographic Channel i 1985, og etter mange år prøvde de å oppsøke henne, uten å finne henne. Det skulle vise seg å være vanskelig å finne henne igjen. National Geographic Channel trengte hjelp fra John Daugman [1]. Daugman er verdenskjent for sitt arbeid med irisgjenkjenning og utviklet de første algoritmene for dette i 1985. Han brukte irismønsteret til kvinnen fra det fotograferte bilde og kvinnens originale irismønster for å identifisere henne.

1.2 Motivasjon

Motivasjonen for oppgaven er å bruke irisgjenkjenning for å verifisere seg ved onlinetjenester som bank og betalingstjenester på nett, slik at dette foregår på en sikker måte. Det blir stadig mer og mer aktuellt å bruke for eksempel mobiltelefonen eller håndholdte enheter til onlinetjenester på internett, og verifikasjon ved hjelp av en persons iris vil være en sikker metode å logge seg inn på slike systemer. Dagens mobiltelefoner og PDA-er har blitt til små personlige datamaskiner integrert med verktøy som er nødvendig for å prosessere og transformere komplekse mediedata. Bilde - og videoprosessering, stemmegjenkjenning, internettsurfing, musikkavspilling, TV og GPS-navigasjon er noen av de

krevende oppgavene disse nå kan brukes til. Derfor vil denne masteroppgaven fokusere på irisgjenkjenning under ugunstige forhold.

Et bruksscenario kan være ved et online banksystem. Banken har lagret en profil av hver bruker bestående av fullt navn, adresse, personnummer, kontonummer og brukerens iriskode. Brukeren kan for eksempel benytte seg av en mobiltelefon, PDA eller sitte foran PC-en med webkamera for å utføre tjenester. Den aktuelle brukeren navigerer seg til innloggingsområdet og vil få følgende melding på skjermen: Tast inn ditt personnummer og trykk på OK knappen. Etter innsending av personnummer vil han få en ny melding på skjermen: Ta bilde av øyet ditt. Brukeren vil da ta bilde av øyet sitt og det vil bli generert en iriskode som overføres til banken. I banken vil det bli foretatt en iriskodevalidering med den lagrede koden som er registrert i brukerens profil. Banken vil godkjenne eller ikke godkjenne innloggingen ut ifra denne iriskodevalideringen. Dersom brukeren får tilgang kan han betale regninger og overføre penger fra en konto til en annen osv. Et online banksystem med irisgjenkjenning som verifikasjonsmetode vil kunne gi en tryggere og mer sikker innlogging enn ved bruk av bare personnummer og pinkode.

Motivasjonen for denne masteroppgaven er å være med å bidra til at fremtidens onlinetjenester blir mer sikre i forhold til inntrengere. Det er i dag veldig mye svindel på internett, og stadig flere benytter seg av VISA og kredittkort på internett for betaling av tjenester. Internett har blitt et verktøy de fleste benytter seg av både ved handel og banktjenester, og irisgjenkjenning for å verifisere brukere vil føre til høy grad av sikkerhet. Med dagens utvikling av informasjons -og kommunikasjonsteknologi er det viktig at brukerne opplever tilgjengelighet og sikkerhet samtidig.

1.3 Forskningsspørsmål

I dette delkapitelet vil forskningsspørsmålene til masteroppgaven bli beskrevet. Det er totalt to forskningsspørsmål som ønskes besvart, og disse fokuserer mot to promblemområder. Det første problemområdet er under hvilke forhold som gjør irisgjennkjenning mulig. Noen viktige faktorer i denne sammenheng er lysintensitet i bildene, uskarphet(blur), støy og rotasjon av bildene. Det andre problemområdet er valg av filteralgoritmer for generering av iriskode. Her vektlegges algoritmenes ytelse under de forskjellige forholdene. En viktig faktor her er også valg av beslutningsterskeler(som avgjør om en får tilgang eller ikke) som bør velges for algoritmene under de forskjellige forholdene. Nedenfor er forskningsspørsmålene beskrevet.

1.3.1 Under hvilke forhold er irisgjenkjenning mulig?

Det første spørsmålet fokuserer på forholdene i det man tar bilde av øyet. Det er flere viktige faktorer som er avgjørende for å besvare dette spørsmålet, men det er blitt gjort en avgrensning i forhold til hvilket det skal eksperimenteres med. Det har blitt valgt å eksperimentere med lysintensiteten i bildet, støyforholdene og uskarpheten som oppstår i bildet, samt stabilitet på hånden i forhold til rotasjon av bildet. Bilder som er tatt med håndholdte kameraer og kameraer som ikke er av beste kvalitet, er ofte utsatt for disse variablene, og dette kan være helt avgjørende når detaljene i irismønsteret skal fanges.

1.3.2 Hvilken filteralgoritme er best under ugunstige forhold?

Algoritmene som velges ut skal implementeres og testes ut under de forskjellige forholdene som er beskrevet over. Det eksisterer mange algoritmer for å filtrere ut detaljene fra irismøsteret og generere iriskode. Ut ifra resultatene fra eksperimentet skal det kunne velges ut et filter som utmerker seg under forskjellige ugunstige forhold. Det skal eksperimenteres med filtere som genererer iriskoder med forskjellig størrelse for å se effekten av antall detaljer fra irisen som kodes. I tillegg skal det ut ifra resultatene kunne bli foreslått beslutningsterskeler for hvert filter under de forskjellige forholdene.

1.4 Avgrensning

Masteroppgaven avgrenses til kun å gjøre eksperimenter i MATLAB med forskjellige algoritmer mot en simulert bildedatabase. Spørsmålet og fokuset i denne oppgaven er å undersøke under hvilke forhold det er mulig å generere iriskode. Fokuset er også rettet mot valg av filteralgoritmer som er best under forskjellige ugunstige forhold.

Algoritmenes ytelse under forskjellige forhold måles og analyseres i MATLAB mot en testdatabase som er manipulert med forskjellige parametere beskrevet i forskningsspørsmål 1. Mer detaljert beskrivelse av algoritmer og parametere det skal gjøres forsøk med, blir beskrevet i kapittel 2 og 3.

1.5 Rapportens struktur

Rapportens innledende sider inneholder sammendrag av oppgaven på norsk og engelsk i tillegg til et forord. Etter dette følger innholdsfortegnelse, figurliste og tabelliste for rapporten.

Kapittel 1 starter med en introduksjon til oppgaven. Kapitlet gir en innledning til problemområdet, samt motivasjon for oppgaven. Kapitlet gir også en beskrivelse av forskningsspørsmålene sammen med avgrensning av oppgaven. Kapitel 2 beskriver teorien som er nødvendig for å forstå problemstillingene som tas opp, samt arbeider som er relatert til oppgaven.

Kapittel 3 tar for seg selve eksperimentet, og hvordan dette er satt opp i forhold til å oppnå resultater som skal danne grunnlaget for å besvare forskningsspørsmålene. Kapittel 4 presenterer resultatene som er oppnådd i eksperimentet, mens kapittel 5 gir en analyse av de oppnådde resultatene.

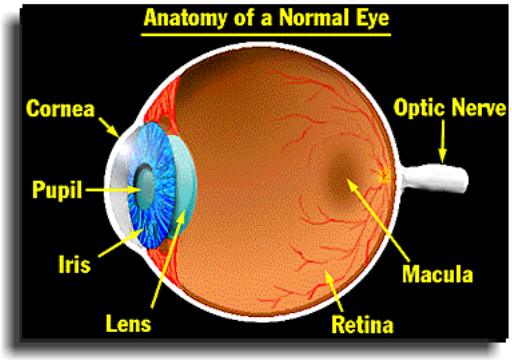
Kapittel 6 oppsummerer og gir en konklusjon på forskningsspørsmålene. Kapittel 7 inneholder forslag til videre arbeid med oppgavens problemområde.

2 Teori og relaterte arbeider

Dette kapitlet tar for seg teorien som er nødvendig for å forstå problemstillingene i oppgaven, samt en beskrivelse av relatert arbeid.

2.1 Øyets oppbygning

Når vi ser på noe, strømmer det lys inn gjennom hornhinnen, den fremste delen av øyet vårt, og videre gjennom linsen. Den former et flatt bilde av omgivelsene på innsiden av øyeeplet, på netthinnen. Der får lyset små følere til å reagere og gi i fra seg signaler som ender opp i hjernen. Hvor tett disse følerne sitter, bestemmer hvor små detaljer vi kan se. I synsgropen, som er det stedet på netthinnen som ser skarpest, er det omtrent 400 reseptorer per millimeter [5]. I figur 2 er det illustrert hvordan øyet vårt er bygd opp.



Figur 2: Illustrasjon av øyets oppbygning.

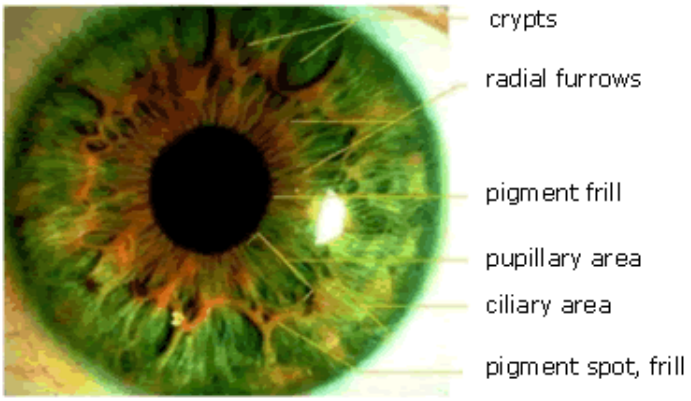
Som vi ser av figuren består øyet av mange forskjellige deler med mange ulike funksjoner. Under kommer en beskrivelse av hver enkelt del i øyet. Videre i denne oppgaven skal vi kun konsentrere oss om irisen.

- Hornhinna (cornea) sitter ytterst på øyet. Hos et menneske har hornhinna en diameter på 11 - 12 mm. Hornhinna har flere viktige oppgaver. Den beskytter den ytre delen av øyeeplet. Sammen med linsa bryter hornhinna lyset.
- Pupillen åpner og slipper igjennom lys. Når øyet utsettes for kraftig lys, trekker pupillen seg sammen. Den fungerer som en blender
- Irisen eller regnbuehinnen er det fargede mønsteret i øyet innenfor det hvite. Vanligvis er den farget brun, grønn eller blå, men det finnes mange forskjellige varianter.
- Linsa bryter lysstrålene sammen med hornhinna slik at det kan dannes et skarpt bilde på netthinnen. Linsa sitter på plass ved hjelp av mange fine fibre. Disse fibrene er festet til muskler som gjør at linsa kan forandre form.

- Synsnerven samler alle signalene som kommer inn i øyet. Den frakter signalene videre opp til hjernen som tolker signalene. Det er hjernen som forteller oss hva vi ser. Om det er noe som ser uklart ut eller vi ikke helt forstår signalene som kommer, prøver hjernen å finne noe likheter med ting vi har sett før. Dette fører til at vi tenker oss hva det er vi ser, før vi egentlig kan være helt sikre på det.
- Netthinnen er et tynt lag bakerst i øyet ditt, som er bygd opp av mer enn millioner ørsmå celler som kan fange opp lys. Når lys treffer en synscelle, sender cellen ut et elektrisk signal. Signalet går til en tynn ledning vi kaller en nervetråd.

2.2 Særtrekk ved menneskets iris

Iris i øyet er et sammentrekkbart membran som er farget, og mønsteret er unikt for hvert enkelt individ. Det finnes ikke to mennesker med lik iris, ikke engang iris på høyre og venstre øye er likt. Derfor er irisgjenkjenning den mest presise og sikre måten å gjenkjenne en person på. Irisen har mer enn 200 punkter [6] som kan trekkes ut og brukes til å sammenligne med andre. Figur 3 viser et eksempel på hvordan et irismønster kan se ut.



Figur 3: Illustrasjon av et irismønster [2]

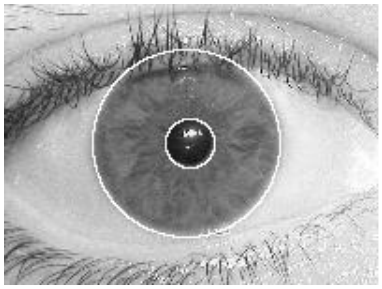
Som figur 3 viser består irisen av mange forskjellige mønster eller særtrekk. De mest atteraktive detaljene i irismønsteret er bl.a kryptene (crypts) som er et relativt tynt området i irisen og som har en veldig mørk farge [6]. Irisen har også detaljer som kalles radiale furer (radial furrows). Disse furene kontrollerer puppilen når den åpner og lukker seg. Et område som kalles pigment strimler (pigment frill) er mønsteret som ligger på grensen mellom pupillen og irisen. Andre detaljer som også er en del av irisen er puppil området, ciliar område og en liten krage(collarette).

2.3 Irisgjenkjennings-prosessen

Prosessen for å gjenkjenne iris består i hovedsak av 4 steg [7]. Disse stegene er segmentering, normalisering, generering av iriskode og sammenligning av iriskoder. Under blir disse stegene beskrevet.

2.3.1 Segmentering

Det første steget i irisgjenkjenningen er å segmentere iris. Dette innebærer å isolere irisregionen i ett digitalt bilde av øyet. Dette gjøres ved hjelp av to sirkler, en for grensen mellom pupillen og iris og en som markerer yttre del av irisen. Figur 4 illustrerer en segmentert iris. Kvaliteten på bildet av øyet har mye å si for resultatet av segmenterin-



Figur 4: Illustrasjon av en segmentert iris

gen. Dersom kontrasten mellom puppilen og irisen er stor, vil segmenteringen gå mye bedre enn hvis kontrasten er liten. Med personer som har brune eller mørke øyer, vil segmenteringen være mye vanskeligere enn med blå eller lyse øyer. Dette skyldes som sagt den lave kontrasten mellom puppilen og iris regionen. Dersom kontrasten blir veldig utydelig kan vi få segmenteringsfeil.

2.3.2 Normalisering

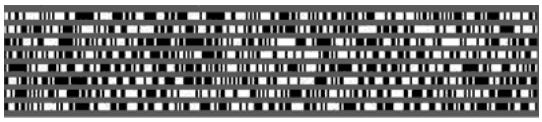
Etter at segmenteringen av iris er fullført, er neste steg å normalisere den segmenterte irisregionen. Da transformeres og brettes irisregionen ut slik at dimensjonene er faste i forhold til iriskodene som skal sammenlignes. Figur 5 illustrerer hvordan irisen er trekt ut og normalisert.

Figur 5: Illustrasjon av en normalisert iris

Dimensjonen på den segmenterte irisregionen varierer på grunn av forskjellige forhold, og normaliseringsprosessen vil da produsere irisregioner som har samme størrelse.

2.3.3 Generering av iriskode

Når normaliseringen er ferdig, blir det generert en iriskode av den segmenterte og normaliserte irisregionen. For å oppnå best mulig presisjon i sammenligningen må den mest signifikante informasjonen trekkes ut fra irisregionen, og denne informasjonen lagres i en iriskode. Figur 6 illustrerer hvordan en iriskode ser ut. En iriskode er en bitrepresen-



Figur 6: Illustrasjon av iris-kode

tasjon av irisen. Koding av slik informasjon kalles feature encoding [3], og det eksisterer mange filtere for å hente ut denne informasjonen. De beste og mest kjente filterene blir beskrevet i neste delkapitel.

2.3.4 Sammenligning

For å sammenligne en ukjent iriskode med andre iriskoder i en database, måles bitdifferansen mellom kodene. Liten bitdifferanse betyr stor sammenheng, og muligens match. For at en sammenligning skal godkjennes som en match, må bit differansen ikke overstige en forhåndssatt beslutningsterskel. En slik terskelverdi er individuell i forhold til filteret og systemet som er implementert.

2.4 Algoritmer for irisgjenkjenning

I dette delkapitlet beskrives forskjellige algoritmer som er utviklet for irisgjenkjenningssystemer. Det eksisterer i dag mange forskjellige systemer, og de beste og mest kjente algoritmene er utviklet av John Daugman [7] [8]. I tillegg til Daugmans system, er det utviklet flere andre systemer som har gitt gode resultater. De beste og mest kjente er systemene til Wildes et al. [9], Boles and Boashash [10], Lim et al. [4], Daouk et al. [11] og Libor Masek [3]. Andre systemer som også er viktige å merke seg er Avila et al. [12], Li Ma et al. [13], Tisse et al. [14], Kong and Zhang [15] og Ritter [16].

2.4.1 Algoritmer for segmentering

Hough-transform

Hough-transform er en teknikk som kan brukes for å isolere særtrekk fra en spesiell figur i et bilde [17]. Den klassiske Houghtransform teknikken er i hovedsak mest brukt for å detektere regulære kurver som sirkler, linjer, ellipser osv. Den sirkulære Hough-transformen brukes til å finne sirkler av en gitt størrelse, og kan bli brukt til å sette radius og senterkoordinater av puppilen og irisregionen. Det er mange som har benyttet seg av denne metoden for segmentering, bl.a Wildes et al. [9], Libor Masek [3],Li Ma et al. [13] og Tisse et al. [14].

Først så genereres en edgemap ved å regne ut den deriverte av intensitetsverdiene i et øyebilde, og deretter terskles resultatet. Denne edgemapen genereres ved å kjøre øyebilde gjennom en såkalt cannyalgoritme som finner kantene i bildet. Ut i fra dette så bestemmes det i Houghrommet parametre for sirkler som passerer gjennom hvert kantpunkt. Disse parameterne er midtpunktkoordinatene x_c og y_c, og radiusen r, som

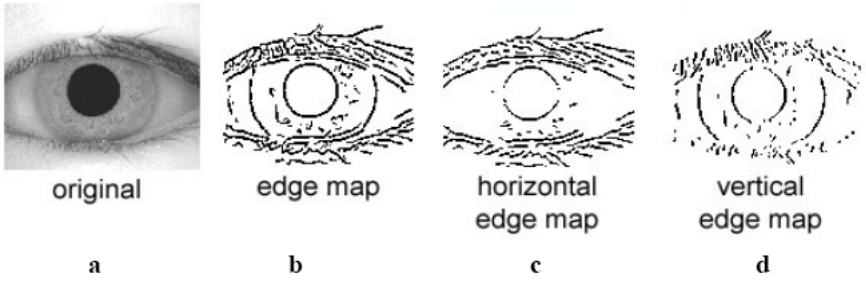
kan definere en hvilken som helst sirkel i henhold til følgende ligning;

$$x_c^2 + y_c^2 - r^2 = 0 \tag{2.1}$$

Et maksimumpunkt i Houghrommet vil stå i sammenheng med radiusen og midtpunktskoordinatene til sirkelen som er best definert av kantpunktene. Wildes et al. [9] og Kong and Zhang [15] benytter også Hough-transformen til å detektere øyelokk. Med en såkalt parabolic Hough-transform detekteres øvre og nedre øyelokk med geometriske buer som er representert ved følgende ligning;

$$(-(x - h_j)\sin\theta_j + (y - k_j)\cos\theta_j)^2 = a_j((x - h_j)\cos\theta_j + (y - k_j)\sin\theta_j). \tag{2.2}$$

Her kontrollerer a_j krummingen på buene, mens (h_j, k_j) er topppunktet på den geometriske parabelen og θ_j er vinkelen på rotasjonen i forhold til x-aksen. For å utføre kantdeteksjon benytter Wildes et al. [9] den deriverte i horisontal retning for å detektere øyelokket, og i vertikal retning for å detektere den ytre grensen av irisen. Figur 7 illustrerer hvordan denne kantdeteksjonen vil se ut.



Figur 7: Illustrasjon av edge map [3] a)Et øye bilde. b)Tilsvarende edge map med både horisontale og vertikale kanter. c)Horisontal edge map. d)Vertikal edge map.

Øyelokket er vanligvis rettet horisontalt og dette gjør det lettere å detektere dem. Edgemap-en med horisontal informasjonen er ikke nødvendig for å detektere iris med vertikale kanter, og med for mye informasjon vil den sirkulære edgemap-en med vertikale data bli ødelagt. Derfor skilles vertikale og horisontale kanter slik at deteksjonen av iris og øyelokk gjøres individuelt. Masek [3] benytter den lineære Hough-transformen til å detektere øyelokk ved trekke en passende horisontal linje til øvre og nedre øyelokk. I steg 2 trekker han en horisontal linje som skjærer den første linjen og iriskanten som er nærmest pupillen. De usignifikante områdene ved øyelokkene ignoreres når irisen trekkes ut for å normaliseres.

Daugman's Integro-Differential Operator

Daugman benytter seg av en integro-differential operator [7] for å segmentere iris -og pupill regionene, samt sirkelbuer for øvre og nedre øyelokk. Integro-differential operatoren er definert som;

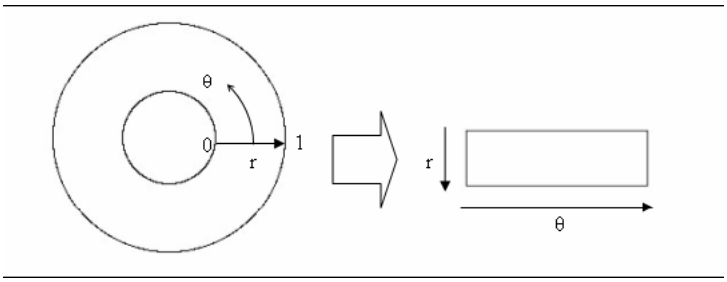
$$max_{(r,x_p,y_0)} \left| G\sigma(r) * \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} ds \right| \quad (2.3)$$

hvor $I(x,y)$ er bilde av øyet, r er radiusen det skal søkes etter, $G\sigma(r)$ er en Gaussisk glattingsfunksjon, og s er omrisset av sirkelen definert av $r, x0, y0$. Operatoren søker etter sirkulære stier hvor det er maksimum forandring i pixelverdier, med å variere radiusen og senter (x, og y) posisjonen av det sirkulære omrisset. Øyelokket blir lokalisert på samme måte, men med sirkelbuer i stede for sirkler. Integro-differential operatoren er en variant av Hough transform. Denne segmenterings metoden beregner også den deriverte av bildet og utfører et søk for å finne de geometriske parametrene.

2.4.2 Algoritmer for normalisering

Daugman's Gummiark modell

Gummiark modellen (Rubber sheet model) av Daugman [7] omadresserer hvert punkt innenfor irisregionen til et par av polare koordinater (r,θ), hvor r er intervallet [0,1] og θ er vinkel $[0, 2\pi]$. Figur 8 illustrerer hvordan dette gjøres.



Figur 8: Illustrasjon av Daugman's gummiark modell [3]

Omadresseringen av irisregionen fra rettvinklede koordinater (x,y) til normaliserte ikke-konsentriske polar koordinater, er beskrevet i følgende uttrykk;

$$I(x(r,\theta), y(r,\theta)) \rightarrow I(r,\theta) \quad (2.4)$$

med

$$x(r,\theta) = (1-r)\, x_p(\theta) + r x_l(\theta) \quad (2.5)$$

$$y(r,\theta) = (1-r)\, y_p(\theta) + r y_l(\theta) \quad (2.6)$$

hvor $I(x,y)$ er irisregionen i bilde, (x,y) er de orginale rettvinklede koordinatene, (r,θ) er de tilsvarende normaliserte polarkoordinatene, og x_p, y_p og x_l, y_l er grensekoordinatene for pupill og iris langs retning θ. Gummiark modellen tar hensyn til pupillutvidelse og uoverensstemmelse i størrelse når det produseres en normalisert representasjon av den segmenterte irisregionen med konstante dimmensjoner. Irisregionen er

modellert som et fleksibelt gummiark med ankerpunkt i irisgrensen, og med pupill som referansepunkt.

Selv om gummiark modellen tar hensyn til pupillutvidelse og bilde distanse, kompenserer den ikke for uoverensstemmelse ved rotasjon. I systemet til Daugman benyttes det rotasjon i løpet av sammenligningen ved å forskyve irismønsteret i retning θ helt til to irismønstre er innrettet. Masek [3] har også implementert gummiark modellen i sitt system for å normalisere irismønsteret.

Bilderegistrering

Systemet til Wildes et al. [9] benytter en teknikk kalt bilderegistrering (image registration), som geometrisk bøyer det segmenterte bildet $I_d(x, y)$ i henhold til et referansebilde i databasen [9]. Når kartleggingsfunksjonen (mapping function) $(u(x, y), v(x, y))$ transformerer orginalkoordinatene, blir intensitetsverdiene i det nye bilde tilsvarende verdiene i referansebildet. Kartleggingsfunksjonen er definert ved;

$$\int_x \int_y \left(I_d\left(x, y\right) - I_a\left(x - u, y - v\right)\right)^2 dxdy \tag{2.7}$$

Transformasjonen av bilde koordinater (x, y) til $(x^{'}, y^{'})$ er definert ved;

$$\begin{pmatrix} x^{'} \\ y^{'} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - sR\left(\phi\right) \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.8}$$

der s er en skaleringsfaktor og $R(\phi)$ er matrisen som representerer rotasjonen med ϕ [9].

Virtual sirkler

Boles and Boashash [10] benytter virtuelle sirkler (virtual circles) ved normalisering. Denne metoden skalerer først irisbildene slik at de har en konstant diameter. Ved sammenligning av to bilder vil det kun bli brukt et referansebilde. Denne teknikken fungerer annerledes enn andre teknikker. Normaliseringen utføres ikke for at to irisregioner skal sammenlignes, men normaliseringen lagres og brukes til sammenligninger senere. Når to iriser har samme dimensjon, blir detaljene i irisen hentet ut fra irisregionen ved å lagre intensitetsverdiene i virtuelle konsentriske sirkler, med referansepunkt i senter av pupillen. Oppløsningen på normaliseringen er konstant så antall punkter som hentes fra hver iris er den samme. Dette er i hovedsak det samme som Daugman gjør i sin gummiark modell.

2.4.3 Algoritmer for generering av iriskode

Wavelet Gabor Filter

Gaborfiltere er i stand til å skaffe en gunstig representasjon av et signal i frekvens rommet [18]. Gaborfilteret er konstruert for å modulere en sinus/cosinus bølge ved en Gaussian. Oppdelingen av et signal gjøres ved bruk av kvadraturpar (quadrature pair) av Gaborfiltere, med en reell del spesifisert av cosinus modulert ved en Gaussian, og en imaginærdel spesifisert av sinus modulert ved en Gaussian [18].

Daugman benytter 2D Gaborfilter [7] for å kode irisregionen. 2D Gaborfilter over et bilde (x, y) er definert som;

$$G(x, y) = e^{-}\pi \left[\left(x - x_0\right)^2 / \alpha^2 + \left(y - y_0\right)^2 / \beta^2\right] e^{\pi i \left[u_0\left(x - x_0\right) + v_0\left(y - y_0\right)\right]} \tag{2.9}$$

hvor (x_0, y_0) spesifiserer posisjonen i bildet, (α, β) spesifiserer bredden og lengden, og (u_0, v_0) spesifiserer modulasjonen som har en romlig (spatial) frekvens $\omega_0 = \sqrt{u_0^2 + v_0^2}$. Daugman demodulerer output fra Gaborfilteret for å komprimere data. Dette blir gjort ved å kvantisere faseinformasjonen inn i fire nivåer, for hver mulige kvadrant i det komplekse planet. Disse fire nivåene er representert med to bits data, slik at hvert pixel i den normaliserte irismønsteret tilsvarer to bits data i iriskoden. Totalt 2048 bits er beregnet for iriskoden, og like mange maskerings bits er generert for å skille ut og ignorere ødelagde områder i irisen. Med dette blir det generert en 256-byte iriskode som kan lagres og sammenlignes med andre iriskoder. I normaliseringen benytter Daugman seg av polarkoordinater, så polar formen av filteret er definert som;

$$H(r, \theta) = e^{-i\omega(\theta-\theta_0)} e^{-(r-r_0)^{2/\alpha^2}} e^{-i(\theta-\theta_0)^2/\beta^2} \tag{2.10}$$

hvor (α, β) spesifiserer bredden og lengden, og (r_0, θ_0) spesifiserer senterfrekvensen av filteret. Demodulasjonen og kvantiseringsprosessen er definert som;

$$h_{\{Re,Im\}} = sgn_{\{Re,Im\}} \int_\rho \int_\phi I(\rho, \phi)\, e^{-i\omega(\theta-\theta_0)} e^{-(r_0-\rho)^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho d\rho d\phi \tag{2.11}$$

hvor $h\{Re, Im\}$ kan bli betraktet som en kompleks vurdert bit hvis real og imaginærkomponentene er avhengig av 2D integralet, og $i(\rho, \phi)$ er det normaliserte irisbilde i et dimensjonsløst polarkoordinatsystem [18].

Wavelet 1D Zero-crossing filter

Boles og Boashash [10] benytter seg av 1D wavelets for koding av irismønster. Utgangspunktet for wavelets er definert som den andrederiverte av glattingsfunksjonen $\theta(x)$ som er definert som;

$$\psi(x) = \frac{d^2\theta(x)}{dx^2} \tag{2.12}$$

Zero-crossing filteret med binær skalering kan brukes til å kode irismønster. Wavelets transformen av et signal $f(x)$ med skalering s og posisjon x kan defineres som;

$$W_s f(x) = f * \left(s^2 \frac{d^2\theta(x)}{dx^2} \right)(x) = s^2 \frac{d^2}{dx^2} (f * \theta_s)(x) \tag{2.13}$$

hvor $\theta_s = (1/s)\,\theta(x/s)$. $W_s f(x)$ er proporsjonal sammenlignet med den andre deriverte av $f(x)$ som er glattet ut med $\theta_s(x)$, og zero-crossing av transformen samsvarer med punktene i bøyningen $f * \theta_s(x)$.

Log-Gabor filter

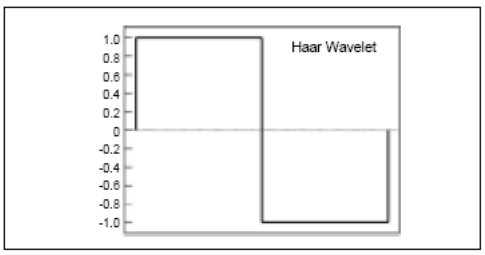
Masek [3] benytter seg av 1D Log-Gabor filter ved koding av irismønster. Den normaliserte irisregionen som er 2-dimensjonal, brytes opp i et antall 1-dimensjonale signaler, og disse 1D-signalene blir kodet med 1D Gabor wavelets. Dette gaborfilteret som er Log (Gaussian on a logarithmic) skalert, er kjent som Log-Gabor filter og er definert som;

$$G(f) = exp\left(\frac{-(\log(f/f_0))^2}{2(\log(\sigma/f_0))^2} \right) \tag{2.14}$$

hvor f_0 representere senter frekvensen, og σ gir frekvensområdet i filteret. Mer detaljert beskrivelse av Log-Gabor filteret er gjort av Field [19].

Haar wavelet

Lim et al. [4] benytter også wavelet transform for å hente ut og kode detaljer i irismønsteret. Både Gabor transform og Haar wavelet kan betraktes som opphavet til wavelet. Haar wavelet benytter flerdimensjonal filtrering. En detaljvektor (feature vector) med 87 dimensjoner blir beregnet. Hver dimensjon har en reell verdi fra -1.0 til +1.0. Figur 9 illustrerer Haar wavelet transformen.



Figur 9: Illustrasjon av Haar wavelet transform [4]

I detaljvektoren er positive verdier representert med 1, og negative verdier med 0. Dette resulterer i en kompakt biometrisk iriskode som består av 87 bits. Mer detaljert beskrivelse av Haar transformen kommer i eksperimentoppsettet.

Laplacian of Gaussian filter

Wildes et.al [9] benytter Laplacian of Gaussian filter for å kode den normaliserte irisregionen. Laplacian of Gaussian filteret er definert som;

$$\nabla G = -\frac{1}{\pi\sigma^4}\left(1-\frac{\rho^2}{2\sigma^2}\right)e^{-\rho^2/2\sigma^2} \tag{2.15}$$

hvor σ er standard avviket av Gaussian og ρ er den radiale distansen av en punkt fra senter av filteret. Det filtrerte bildet er representert som en Laplacian pyramide som kan komprimeres slik at bare signifikante data blir igjen. Laplacian pyramiden er konstruert med fire forskjellige oppløsningsnivåer for å generere en kompakt iriskode.

2.4.4 Algoritmer for sammenligning

Hammingdistanse

Hammingdistanse er en metode for å måle avstand mellom to datamengder, som bl.a Daugman [7] og Masek [3] har implementert. Ved sammenligning av to iriskoder, KodeA og KodeB, kan KodeA være lagret i databasen og KodeB være den genererte iriskoden. Hamming distansen beregnes ut fra følgende formel;

$$HD = \frac{\|(KodeA \otimes KodeB) \cap MaskA \cap MaskB\|}{\|MaskA \cap MaskB\|}, \tag{2.16}$$

hvor MaskA og MaskB brukes for å luke bort usignifikante bit. Operasjonen $\otimes$ gir en XOR sammenligning av kodeA og kodeB, som skal finne ulikheten mellom iriskodene. Resultatet fra denne operasjonen gir 1 dersom to bit er ulike, og 0 dersom de er like. Hvert bit fra resultatet av denne XOR operasjonen skal sammenlignes med maskene til A og B. Til dette brukes AND operasjonen $\cap$. Dette gjøres som sagt for å luke vekk usignifikante bit som refleksjoner, øyevipper og øyelokk etc. Et bit er med bare hvis AND operasjonen mellom kodene og maskene er sann. For å summere sammen de bitene som er sanne benyttes operasjonen $\|\|$.

Telleren i brøken gir svar på hvor mange bit som er ulike av de signifikante fra A og B. Nevneren finner lengden på antall bit som er signifikante, altså bitene som skal være med i vurderingen av Hammingdistansen. Jo lavere Hammingdistansen er, jo mer sikkert er et gyldig treff. Dersom Hammingdistansen er 0, er det en helt perfekt match.

Weighted Euclidean Distance

Weighted Euclidean Distance (WED) er også en metode for å sammenligne to iriskoder, spesiellt hvis iriskoden består av heltallsverdier. WED gir et mål på hvor lik en samling av verdier er mellom to iriskoder. WED beregnes ut fra følgende formel;

$$WED(k) = \sum_{i=1}^{N} \frac{\left(f_i - f_i^{(k)}\right)^2}{\left(\delta_i^{(k)}\right)^2} \tag{2.17}$$

hvor f_i er det i^{th} særtrekket(et bit) av den ukjente iriskoden, $f_i^{(k)}$ er det i^{th} særtrekket av den lagrede iriskoden og k og $\delta_i^{(k)}$ er standardavviket av i^{th} særtrekket i den lagrede iriskoden k. Den ukjente iriskoden er funnet som en match med iriskode k, når WED er et minimum av k.

Normalised korrelasjon

Normalisert korrelasjon (Normalised correlation) er en sammenligningsmetode Wildes et al. [9] har implementert. Metoden går ut på å finne en normalisert korrelasjon mellom den ukjente iriskoden og den representerte iriskoden i databasen. Metoden er definert følgende;

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(p_1[i,j]-\mu_1\right)\left(p_2[i,j]-\mu_2\right)}{nm\sigma_1\sigma}{}_1 \tag{2.18}$$

hvor p_1 og p_2 er to iriskoder med størrelse $n \times m$, μ_1 og σ_1 er gjennomsnittet og standardavviket av p_1 og μ_2 og σ_2 er gjennomsnittet og standardavviket av p_2. Normalisert korrelasjon er fordelaktig med standardavvik, siden den kan ta hensyn til lokale variasjoner i bildeintensiteten.

2.5 Relatert arbeid

Det har blitt utviklet mange algoritmer for irisgjenkjenning. Mange av disse algoritmene er imponerende bra og leverer gode resultater. Som nevnt tidligere i kapittel 2, er de beste og mest kjente algoritmene utviklet av John Daugman [7]. Daugman har implementert integro-differential operator for segmentering, gummiark modell for normalisering, wavelets 2D Gabor-filter for generering av iriskode og Hammingdistanse for sammenligning av to iriskoder. På en 300 MHZ Sun workstation sammenligner Daugman's system 100.000 iriskoder i sekundet. Det har blitt gjort 9.1 millioner sammenligninger mellom øyne fra Storbritania, USA, Japan og Korea. Av disse sammenligningene identifiserer systemet korrekt i 99.9 prosent av tilfellene. Med statistisk beregning har Daugman funnet ut at to iriser som produserer en Hammingdistanse HD ≤ 0.32 for dette systemet, må være en korrekt match. Med denne beregningen er oddsen for feil match 1 til 26 millioner. Dette er utgangspunktet for irisgjenkjenning.

Masek [3] utviklet en opensource kode med irisgjenkjenning i sin bacheloroppgave ved universitetet i Vest-Australia 2003. Det har blitt implementert Hough-transform for segmentering, gummiark modell for normalisering, 1D Log-Gabor filter for generering av iriskode og Hammingdistanse for sammenligning av iriskoder. Masek har testet systemet sitt på to sett med bilder, et med 75 bilder fra LEI databasen og et med 624 fra CASIA databasen. På settet med 75 bilder gjenkjenner systemet perfekt med False Accept og False Reject prosent lik 0. På det andre settet med 624 bilder ble også resultatene meget gode, med False Accept og False Reject på henholdsvis 0.005 prosent og 0.238 prosent. Totalt sett så er dette systemet meget bra, og med bakgrunn i eksperimentresultater konkluderer Masek med at irisgjenkjenning er en pålitelig biometrisk teknologi med med høy presisjon.

I systemet til Wildes et al. [9] har det blitt implementert Hough-transform for segmentering av iris og øyelokk, image registering for normalisering, Laplacian of Gaussian filter for generering av iriskode og til sammenligningen av to iriskoder er det implementert en metode som heter normalisert korrelasjon. Dette systemet har levert veldig gode resultater, men har ikke blitt testet i samme grad som systemet til Daugman. Det har blitt testet med totalt 60 forskjellige iriser fra 40 personer. Fra hver iris har det blitt testet med 10 bilder, 5 fra begynnelsen av eksperimentene og 5 fra en måned senere. Testsettet inkluderte også iriser fra eneggede tvillinger, og med forskjellig farge på iris (blå, grønn, brun og nøttebrun). I eksperiment resultatene har det ikke blitt observert false positives eller false negatives med dette testsettet. Som sagt så er testsettet veldig lite i forhold til settet som har blitt testet i systemet til Daugman, men alikevel så har systemet levert veldig gode resultater.

Boles og Boashash [10] har utviklet et irisgjenkjenningsystem som skal takle forskjellige forhold. Dette er forhold med forskjellig støy -og lysforhold, og med varierende avstand fra øyet. Systemet baseres på gråskalabilder og lager en iriskode ut fra disse. Dette gjøres ved at 1D signaler hentes ut av det forhåndsprosesserte irisbilde og disse signalene representeres med zero-crossing filter. Segmenteringen gjøres med en kantdeteksjonsalgoritme, som setter sirker rundt irisregionen med referansepunkt i pupillen. Ved normalisering har Boles og Boashash implementert en metode som heter virtuelle sirkler (virtual

circles). For å sammenligne benyttes en ulikhetsfunksjon som beregner ulikheten mellom to iriskoder. I eksprimentdatabasen har det blitt brukt kun et bilde fra hver person (venstre eller høyre). I eksprimentene har det blitt gjort søk med to like bilder som er tatt under forskjellig lysforhold og fra forskjellig distanse mellom kamera og øyet. Resultatet av dette var at ulikheten mellom disse to bildene var mye mindre enn med de andre det ble søkt gjennom. Dette beskriver Boles og Boashash som bra irisgjenkjenning mellom forskjellige irismønster. I et annet eksperiment ble det gjort tilsvarende forsøk, med 11 bilder i databasen bestående av forskjellige iriser med forskjellig oppløsning. Irisene det søkes med er to iriser av samme øye, bare med forskjellig lyssetting. Effekten av støy ble også studert, der bildene ble kjørt gjennom et medianfilter før prosessering av iriskode. Det ble eksprimentert med et signalstøy forhold (SNR) fra 0 til 30 dB. Resultatene her er også gode, men systemet feiler litt ved lavere SNR mellom de to bildene det søkes med. Forskjellen mellom disse bildene er som sagt bare forskjellig lysforhold, og lavt støynivå i bildene gjør at systemet feiler noe. Totalt så leverer systemet veldig gode resultater, men det må påpekes at eksprimentene har blitt kjørt mot en veldig liten database.

Lim et al. [4] har implementert et system som baserer seg på Haar wavelet. For segmentering benyttes en kantdeteksjon for så å finne ytre og indre grense av irisen, som markeres med to sirkler. Normaliseringsprosessen er i stor grad den samme som Daugman [7] har implementert, ved at alle detaljene i irisen transformeres til det polare koordinatsystemet. Detaljene i irisen transformeres til en normalisert iriskode i henhold til vinkel og radius (θ, r) i det segmenterte bildet. For å generere iriskode er det som sagt implementert Haar wavelet for å kode detaljene i den normaliserte iristemplaten. Det har blitt implementert to størrelser på irisvektoren. Den ene er på 87-bits, og den andre er på 2048-bit. Ytelsen på disse to størrelsene har blitt evaluert og resultatene viser at vektoren på 87-bits gir noe bedre match prosent. For sammenligning av iriskoder har det blitt implementert en medtode som heter multi-dimensional winner selection (for detaljert beskrivelse, se [4]), som i stor grad er en optimalisering av WED (Weighted Euclidean Distance) metoden. I eksperimentene har systemet blitt testet på et stort sett med data som inneholder 6000 bilder. I alt har det blitt eksperimentert med 200 koreanske universitetsstudenter hvor det har blitt brukt 30 bilder av hver. Disse bildene er tatt i løpet av en tre måneders periode. Eksperimentene ble delt inn i to deler, den første delen tok for seg ytelsen til det implementerte systemet, mens den andre delen tok for seg FAR (False accept rate) og FRR (False reject rate) i forhold til match prosent. I del en er det gjort en sammenligning av Gabor og Haarfilter, der Haar wavelet filter viser seg å være 0.8 og 2.1 prosent bedre på to forskjellige datasett. En sammenligning av WED og Multi-dimensional viser at Multi-dimensjonal gir noe bedre resultat, med henholdsvis 0.7 og 1.3 prosent bedre på samme datasett. Eksperimentet viser at systemet har en total match prosent på 98.4 prosent, som er veldig bra. I del to av eksperimentet ble FFR og FAR undersøkt. Systemet viser seg å gi lave FFR og FAR ved bruk av høye terskelverdier når to iriskoder sammenlignes(se [4]). Daouk et al. [11] har også implementert et irisgjenkjennings system basert på Haar wavelet. I likhet med Lim et al.[4] blir Haar wavelet filteret benyttet til å filtrere ut og kode polarkoordinatene fra den normaliserte iristemplaten. Sammenlignet med Lim et al. genererer Daouk et al. en iriskode(irisvektor) på 702-bit. Flere detaljer fra Haar transformen kombineres for å representere iriskoden (se [11]).

Vatsa et al. [20] har i sin artikel studert og undersøkt fire iris gjenkjenningssystemer. Dette er algoritmer som er utviklet av Avila et al. [21], Li Ma et al. [13], Tisse et al. [14] og John Daugman [7]. Vatsa et al. har implementert alle fire systemene og sammenlignet resutatene. Systemet til Avila et al. er i stor grad det samme som Boles og Boashash [10] med wavelet zero-crossing. Li Ma et al. har utviklet et system som baseres på sirkulære symetriske filter [13], mens i systemet til Tisse et al. er det implementert et nytt konsept som heter instantaneous-phase (se [14]). Systemet til Daugman baserers på 2D Gabor wavelet. I dette eksprimentet ble det opprettet en database med 756 iriser fra 108 personer, hvorav 7 bilder fra hver person. Resultatene fra eksperimentet viste at algoritmene til Daugman ga best resultater med FAR/FRR på 0.01/0.09. Systemet til Li Ma hadde FAR/FRR på 0.02/1.98 mens systemet til Avila hadde en FAR/FRR på 0.03/2.08. Til slutt i sammenligningen kom systemet til Tisse med FAR/FRR på 1.84/8.79.

På ITU Telecom World messen i Hong Kong i slutten av 2006, introduserer OKI [22] Japan's første teknologi for irisgjenkjenning på mobiltelefoner. Denne teknologien er beregnet for bilder som er tatt med en mobiltelefonen eller håndhold enhet, og baserer seg på standardalgoritmene til OKI. Teknologien lanseres i første halvdel av mars 2007 og resultatene er også gode, med bare 1 av 100.000 forsøk som gir feil [22]. Cho et al. [23] har i sin artikkel diskutert behovet for sikre løsninger på mobiltelefoner i forbindelse med banktransaksjoner og tjenester via telefon. De har da foreslått segmenteringsalgoritmer for å lokalisere iris på mobiltelefon som både skal være raske og effektive. Disse algoritmene er utviklet spesiellt for å takle forhold som kan oppstå under slike forhold, som forandring av lysintensitet og kontrast som kan gjøre bilde uskarpt. Algoritmene er også uviklet slik at de er tilpasses prosessorkraften som er i dagens mobiltelefoner. Algoritmene er blitt testet under to forskjellige eksperimenter, der det første eksperimentet gikk på prosesseringstiden av algoritmene, der disse bruker 160 msek på å lokalisere iris sammenlignet med algoritmene til Daugman som bruker 340 msek. I eksperiment to ble algoritmenes presisjon målt. Det ble brukt 1000 testbilder som ble tatt med en Samsung SPH-S2300 telefon. Resultatene viser omtrent samme presisjon og ytelse som algoritmene til Daugman samtidig som proseseringstiden er mye bedre. Resultatene viser at de foreslåtte algoritmene egner seg til sanntids irislokalisering for irisgjenkjenning på mobil og håndholdte enheter.

3 Eksperimentoppsett

Dette kapitlet inneholder en beskrivelse av hvordan eksprimentet er satt opp og gjennomført. Kapitlet gir en beskrivelse av testkoden som er brukt i eksperimentet, tilpassing av koden samt valg og implementasjon av filteralgoritmer som har blitt testet i eksprimentet. I tillegg til dette er oppbyggingen av irisdatabasen beskrevet med hvordan irisbilder har blitt manipulert med forskjellige parametere for å skille egenskapene til de ulike filterene. Til slutt gir kapitlet en beskrivelse av evalueringen og oppbygningen av dataanalysen i tillegg til hvordan resultatene er registrert.

3.1 Tilpassing av testkode

Eksprimentene har foregått i matematikkprogrammet MATLAB, og det har blitt tatt utgangspunkt i open source koden til Masek [3]. Selve segmenterings og normaliserings algoritmene har det ikke blitt gjort noen forandringer med da dette ikke var en del av oppgaven. I den delen av koden som filtrerer ut detaljer i irisen og genererer iriskode av den normaliserte iristemplaten, har det blitt implementert tre filter i tillegg til Log-Gabor filteret som er original filteret i koden. Hvilke filter som er implementert og bakgrunn for valgene er diskutert i neste delkapittel.

MATLAB koden til Libor Masek har 2 hovedfunksjoner, CREATEIRISTEMPLATE og GETHAMMINGDISTANCE. CREATEIRISTEMPLATE funksjonen kaller andre nødvendige funskjoner for å segmentere iris, normalisere iris og kode iristemplaten med 1D Log-Gabor filter. Funksjonen GETHAMMINGDISTANCE beregner Hammingdistansen mellom to iriskoder med tilhørende støymasker, og returnerer kun en heltalls verdi mellom 0 og 1.

3.1.1 Tilpassing av Createiristemplate

Etter å ha studert og forstått koden til Libor Masek, ble koden tilpasset eksperimentene som skulle kjøres. Som nevnt over har det ikke blitt gjort forandringer i algoritmene for segmentering og normalisering. Utgangspunktet for CREATEIRISTEMPLATE var at funksjonen hadde en inputparameter. Denne input parameteren var et irisbilde. For å automatisere eksperimentet mest mulig ble det i denne funksjonen implementert en multifunksjon som leser inn alle bildene i testdatabasen samtidig. Disse bildene segmenteres, normaliseres og kodes i den rekkefølgen de leses inn. Til slutt blir arrayen av hver iriskode med tilhørende støymaske lagret som en MATLAB data fil i en egen katalog. Alle iriskoder og støymasker får automatisk en unik id fra 1 og oppover i samme rekkefølge som de leses inn.

3.1.2 Tilpassing av Gethammingdistance

Utgangspunktet for funksjonen GETHAMMINGDISTANCE var at den hadde fire input parametere, iriskode med tilhørende støymaske for to personer. For å sammenligne iriskoder har det i denne funksjonen blitt implementert egne sammenligningsalgoritmer for å automatisere eksperimentet mest mulig. Det ble implementert to forskjellige sammenligningsalgoritmer, en for interklasse-sammenligning og en for intraklasse-sammenligning. Interklasse-sammenligning gjøres for å eksperimentere med FAR (false acsept rate), som sier noe om hvor mange som får tilgang som ikke skal ha det. Intraklasse-sammenligning gjøres for å eksperimentere med FRR (false reject rate), som sier noe om hvor mange som ikke får tilgang som skal ha det. Begge sammenligningsargoritmene leser alle arrayene med iriskoder inn samtidig. All sammenligning skjer automatisk avhengig av om det er interklasse eller intraklasse-sammenligning som velges. Ved interklasse sammenlignes alle iriskodene med unntak av de som er av samme person. Ved intraklasse sammenlignes alle iriskodene av samme person. Verdiene fra sammenligningene legges i en array. For å se hvordan Hammingdistansene fordeler seg, plottes alle verdiene i et stolpediagram. Til slutt har det blitt implementert en automatisk beregning av FRR og FAR ut ifra antall Hammingdistanser ved forskjellige terskelverdier.

3.2 Implementering av filter

Etter å ha nøye studert de forskjellige filteralgoritmene for å generere iriskode, ble det gjort et valg for hvilke filter som skulle testes i eksperimentet. Siden koden til Libor Masek i stor grad er implementert etter systemet til Daugman med Gabor filter, var det naturlig å ha med dette filteret i eksperimentet. MATLAB koden for dette filteret var også ferdig implementert. Systemet til Wildes et al. [9] er sammen med systemet til Daugman det mest kjente av alle irisgjenkjennings systemer, derfor ble det valgt å implementere Laplacian of Gaussian filter til eksperimentet. Det ble naturlig å velge filter som genererer mindre iriskoder, derfor ble det valgt å implementere Lim et al. [4] sin 2D Haar wavelet filter som kun resulterer i en 87-bits irisvektor. For å se effekten av hvor stor iriskoden er og hvor mange detaljer som kodes, ble det i tillegg til dette filteret implementert en større og utvidet iriskode med 2D Haar wavelet. En 702-bit Haar wavelet filter ble implementert på samme måte som Daouk et al. [11] har implementert det. Ettersom disse Haar wavelet filterene genererer iriskoder med langt færre bits, er det ikke nødvendig med støymasker, derfor har det blitt implementert en egen Hammingdistanse funksjon som tar hensyn til dette. Denne funksjonen er i stor grad den samme som Libor Masek har implementert. Hovedforskjellen er at den ikke trekker fra de usignifikante bit-ene som ligger i støymaskene. Under er det beskrevet hvordan de utvalgte filterene er implementert.

3.2.1 Log-Gabor og Laplacian of Gaussian

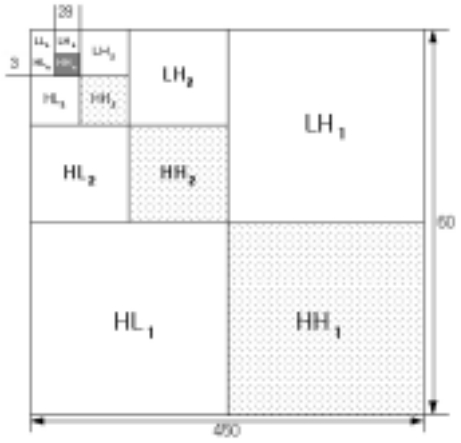
Masek har etter flere forsøk funnet ut at den beste normaliserings-størrelsen av iristemplaten er 240x20. Log-Gabor filteret genererer iriskoder på 9600-bit med tilhørende støymasker på 9600-bit av den normaliserte iristemplaten som består av polarkoordinater. Med en normalisert størrelse på 240x20 resulterer dette i en iriskode på 480x20 på grunn av de returnerte verdiene som består av komplekse tall. Disse komplekse tallene kodes på annenhver linje. På oddetalls linjer kodes realdelen, mens på partalls linjer kodes imaginærdelen av de returnerte filtrerte verdiene. Alle komplekse tall konverteres til 0 eller 1 ettersom om de er positive eller negative. Støymasken blir generert ut ifra tall som har en absoluttverdi mindre eller lik 0.0001. Disse tallene er usignifikante og betraktes som støy og konverteres til 1. Resten av støymasken har 0 verdier. Dette resulterer i en iriskode på 9600-bit med en tilhørende støymaske på 9600-bit.

Laplacian of Gaussian filteret har blitt implementert på samme måte som Libor Masek har implementert Log-Gabor filteret, med en normalisert iristemplate på 240x20. Samme metode har blitt brukt her for å generere iriskode av de komplekse verdiene, som resulterer i en iriskode på 9600-bit med tilhørende støymaske. Forskjellen mellom Log-Gabor og Laplacian of Gaussian er metoden for hvordan iristemplaten blir filterert, som er beskrevet i teorikapitelet. I begge filterene benyttes det senterfrekvenser på 0.5. Begge filterene opererer på endimensjonale vektorer, så den normaliserte iristemplaten deles opp i endimensjonale vektorer (rader) som transformeres fra tidsplanet og over til frekvensplanet før filtreringen skjer. Etter filtreringen transformeres den filtrerte vektoren over til tidsplanet igjen. Når alle vektorene i den normaliserte iristemplaten er filtrert, settes de sammen igjen todimensjonalt før alle de koplekse filterverdiene konverteres til reelle verdier, 0-ere og 1-ere.

3.2.2 Haar wavelet 87-bits vektor

For å oppnå en iriskode(irisvektor) på 87-bit, normaliseres den segmenterte irisregionen til en størrelse på 450x60. Disse variablene settes i funksjonen Createiristemplate, og sendes med videre til funksjonen Normaliseiris som normaliserer iristemplaten. Den normaliserte iristemplaten som består av polarkoordinater kjøres gjennom Haar transformen fire ganger som resulterer i fire delbilder med en størrelse på 28x3. For hver gang Haar transformen filtrerer, halveres bildet både i horisontal og vertikal retning. Dette resulterer i et bilde som har blitt høypass filtrert i horisontal og vertikal retning, et som har blitt lavpass filtrert i vertikal retning og høypass filtrert i horisontal retning, et som har blitt høypass filtrert i vertikal retning og lavpass filtrert i horisontal retning og til slutt et som har blitt lavpass filtrert i både horisontal og vertikal retning.

Figur 11 viser et konseptuelt diagram for hvordan irisvektoren blir konstruert. For å organisere irisvektoren på 87-bit kombineres kun de høypass filtrerte detaljene som ligger diagonalt.

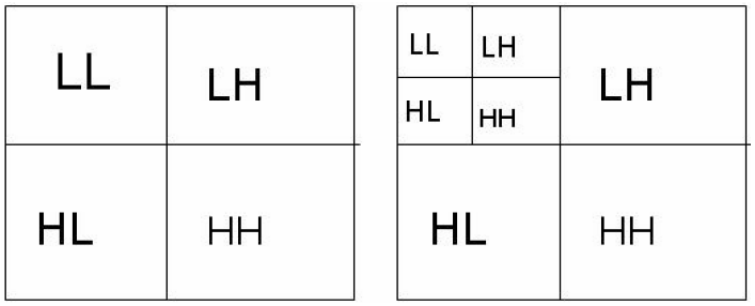


Figur 10: Konseptuelt diagram for 87-bits Haar wavelet[4]

Denne vektoren dannes ved å kombinere de høypassfiltrerte detaljene fra den fjerde Haar transformen, sammen med gjennomsnittet fra de høypassfiltrerte fra den første, andre og tredje transformen. Vektoren består nå av reelle tall mellom -1.0 og 1.0. For å oppnå en 87-bit vektor konverteres alle negative verdier til 0, og alle positive til 1.

3.2.3 Haar wavelet 702-bit vektor

For å oppnå en vektor på 702-bit, normaliseres den segmenterte irisregionen til en størrelse på 402x100. Prinsippet er her i hovedsak det samme som ved genereringen av 87-bit vektor, men her kjøres den normaliserte iristemplaten med polarkoordinater gjennom Haar transformen fem ganger. I denne vektoren hentes det mange flere detaljer sammenlignet med vektoren på 87-bits. For å organisere en irisvektor på 702-bit, hentes alle de returnerte matrisene fra den fjerde og den femte transformen ut for å generere vektoren. Disse matrisene representere koeffisientene fra vertikal informajson, horisontal informajson og diagonal informajson på fjerde og femte nivå.

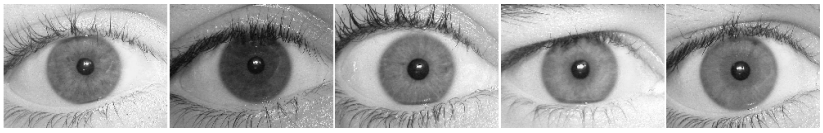


Figur 11: Haar wavelet for organisering av 702-bit vektor

Figur 11 illustrerer hvordan irisvektoren organiseres. Figuren til venstre viser den første Haar transformen. Figuren til høyre viser den andre Haar transformen der det returnerte bilde som er lavpassfiltrert i begge retninger fra den første transformen, filtreres på nytt. Denne filtreringen gjøres på samme måte i den tredje, fjerde og femte transformen. Input i disse Haar transformene er det lavpassfiltrerte bilde fra forrige transform, med unntak fra første transform som filtrerer den normaliserte iristemplaten. På samme måte her konverteres alle negative tall til 0 og positive tall til 1. Dette resulterer i en iriskode/vektor på 702-bits.

3.3 Test database

Det finnes flere tilgjenglige irisdatabaser som kunne lastes ned og brukes i eksperimentet. CASIA databasen [24] er den mest brukte databasen med krystall klare bilder av meget god kvalitet. LEI databasen [25] er også av samme karakter med bilder av meget god kvalitet. Til eksperimentene har det blitt valgt å bruke UBiris databasen [26]. Denne databasen består av 1205 bilder av 241 personer, altså 5 bilder av hver person. UBiris databasen kan lastes ned med 3 forskjellige kvalitetsnivåer og formater. Disse formatene er 800x600 24 bit farge, 150x200 24 bit farge og 150x200 gråskalabilder. Irisbildene i farge med størrelse 800x600 er av veldig god kvalitet, og disse ble derfor ikke valgt til eksperimentet. For å gjøre eksperimentet mest mulig realistisk, ble det valgt å bruke gråskalabildene med format 150x200. Her er kvaliteten forskjellig og bilder av samme person kan variere mye. Det er forskjellig hvor mye av irisen som er med siden øyelokkene kan hindre i å trekke ut tilstrekkelig informasjon. Figur 12 viser fem forskjellige bilder fra irisdatabasen.

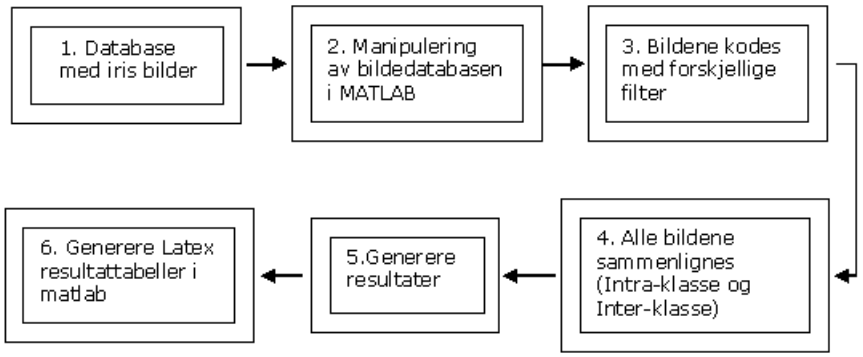


Figur 12: Eksempler på gråskalabilder fra UBiris databasen

Det har blitt gjort et tilfeldig utvalg fra UBiris databasen [26] for å lage en testdatabase til eksperimentet. Testdatabasen er begrenset til 100 personer, hvorav fem bilder fra hver person. Dette resulterer i en testdatabase på totalt 500 bilder. Denne begrensningen er gjort med hensyn til tiden det tar å generere iriskoder, samt å sammenligne alle iriskodene.

3.4 Eksperimentet

Hensikten med eksperimentet er å se hvordan irisgjenkjenningen påvirkes av forskjellige endringer av parametere i testdatabasen, og hvilket filter som gir best resultat ved de forskjellige forholdene. Dette er parametere som typisk kan oppstå på bilder som er tatt med kameraer som ikke er av beste kvalitet. Eksempler på slike er kamera på mobiltelefon, PDA, webkamera og lignende. Det har blitt gjort et utvalg av parametere som bildene skulle simuleres etter. Disse paramentrene har blitt endret ved å legge til forskjellig nivåer Gaussian støy og uskarphet (Gaussian blur), forandring av lysintensitet og rotasjon av bildene. Figur 13 illustrerer hvordan eksperimentet har blitt gjennomført.

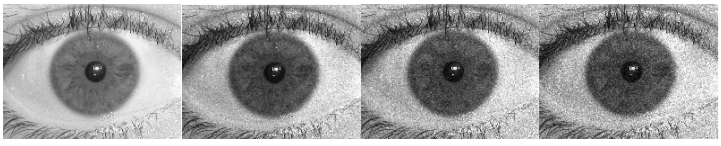


Figur 13: Illustrasjon av hvordan eksperimentet har blitt gjennomført

Utgangpunktet er testdatabasen uten forandringer. Bildene manipuleres med forskjellige paramentere (disse blir beskrevet under i de neste delkapitlene). Etter at testdatabasen er manipulert, kodes de normaliserte iristemplatene med 702-bit og 87-bit Haarfilter, Log-Gabor og Laplacian of Gaussian filter med 9600-bit kode. For hvert filter og hver endring av parameter har det blitt gjort to forskjellige sammenligninger for å skille egenskapene til filtrene. For å eksperimentere med False Acsept Rate (FAR) har det blitt gjort interklasse-sammenligninger. Ved interklasse-sammenligning blir det sammenlignet totalt 123750 for hvert filter og hver parameter som endres. For å eksperimentere med False Rejection Rate (FRR) har det blitt gjort intraklasse-sammenligninger. Totalt så sammenlignes alle fem iriskodene med hverandre, som resulterer i ti sammenligninger for hver person. Dette gir 1000 intraklasse-sammenligninger for alle personene for hver parameter som endres.

3.4.1 Eksperiment med støy

For å eksperimentere med støy i testdatabasen, har det blitt lagt til Gaussian støy på alle bildene. På bakgrunn av en pretest ble det lagt til Gaussian støy med forskjellige støyvarianse. Denne pretesten viste at en støyvarianse på 0.008 ga feil i segmenteringsprosessen på mange av bildene. Med dette har det blitt eksperimentert med en støyvarianse i bildene på 0.002, 0.004 og 0.006. Dette ble gjort med følgende matlabkode for hvert bilde som leses inn og kodes: $imnoise = (gaussian, m, v)$ der støyvariansen settes i parameter v. Figur 14 viser hvordan et bilde fra databasen forandrer seg ettersom det blir manipulert med de forskjellig nivåene av Gaussian støy.



Figur 14: Bildet til venstre er originalbildet, mens de etterfølgende er manipulert med gaussisk støy nivå på 0.002, 0.004 og 0.006

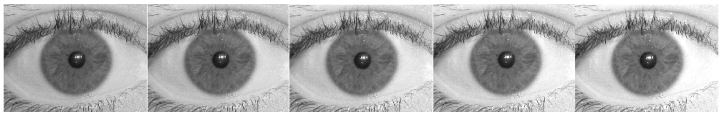
3.4.2 Eksperiment med uskarphet

Det har også blitt eksperimentert med uskarphet(blur) i bildedatabasen med forskjellig nivå (blur radius). Det ble utførte en pretest for å kartlegge hvor mye blur som skulle legges til i bildedatabasen for å skille hvem filter som ga best resultat. Av pretesten ble det bestemt å legge til blur med radius 2, 4, og 6 siden 8 ga sementeringsfeil på mange bilder. Dette ble gjort med følgende matlabkode for hvert bilde som leses inn og kodes: $\mathrm{H} = \mathtt{fspecial}(\mathtt{'disk'}, \mathtt{r})$; der r parameteren spesifiserer radiusen med blur. For å legge til dette i hvert bilde ble følgende kode kjørt: $\mathtt{im} = \mathtt{imfilter}(\mathtt{im}, \mathrm{H}, \mathtt{'replicate'})$; der aktuellt bilde får lagt til blur som ble spesifisert i variabel H. Figur 15 viser hvordan et bilde fra databasen forandrer seg ettersom det blir manipulert med forskjellig radius av blur.

Figur 15: Bildet til venstre er original bilde uten forandring. De etterfulgte bildene har blitt lagt til blur med radius 2, 4 og 6.

3.4.3 Eksperiment med lys

For å eksperimentere med lysforandring i bildedatabasen har lysintensiteten både blitt økt og redusert i bildene. En pretest viste at det ikke er mye forandring som skal til før resultatene endrer seg veldig mye sammenlignet med resultatene ved optimale forhold uten forandring. For å kartlegge hvilke verdier det skulle eksperimenteres med, ble lysintensiteten i pretesten økt og redusert mellom 20 og 50 prosent. Dette viste seg å være for mye for å sammenligne ytelsen til de forskjellige filtrene. Ut ifra pretesten og tidsbegrensninger ble det valgt å eksperimentere med å redusere og øke lysintensiteten i bildedatabasen med 5 og 10 prosent. Figur 16 viser et bilde som blir manipulert med forskjellige endringer i lysintensiteten.

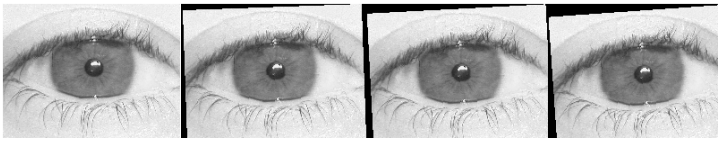


Figur 16: Til venstre i figuren er lysintesiteten redusert med 10 og 5 prosent. Bildet i midten er original bilde, mens til høyre er lysintensiteten økt med 5 og 10 prosent.

Dette ble gjort direkte i matlab ved å gange opp eller ned matrisene til hvert bilde ettersom de ble lest inn for å segmenteres og kodes. Ved å redusere lysmengden i bildene ble matrisene ved 10 prosent intensitets reduksjon ganget ned med 0.90, og ved 5 prosent reduksjon ganget ned med 0.95. Det samme ble gjort for å øke lysintensiteten ved å gange bildene opp med 1.05 og 1.10 for å oppnå 5 og 10 prosent økning.

3.4.4 Eksperiment med rotasjon

Til slutt ble det eksperimentert med rotasjon av bildene i testdatabasen. En pretest viste at det ikke er mye som skal til før det oppstår feil i segmenteringen. Rotasjon på mellom 5 og 10 grader viste seg å være for mye til at alle bildene kunne segmenteres uten feil. Ut ifra denne pretesten gikk det greit å segmentere bilder som har blitt rotert mindre en 5 grader, derfor har det for hvert filter blitt eksperimentert med en testdatabase der bildene har blitt rotert 2, 3 og 4 grader. Dette ble gjort med følgende kode: $B = imrotate(im, angle)$; der rotasjonsgraden settes i parameteren angle. Figur 17 viser hvordan et bilde har blitt rotert med forskjellige grader.



Figur 17: Bildet til venstre er original bildet uten rotasjon. De etterfulgte er rotert med 2, 3 og 4 grader.

3.5 Evaluering og registrering av resultater

For å besvare forskningsspørsmålene må resultatene evalueres. Det har blitt generert mange resultater som må tolkes før det kan konkluderes med noe. For å oppsummere eksperimentet så har det altså blitt kjørt test med fire forskjellige filteralgoritmer mot en testdatabase på 500 bilder. Testdatabasen har blitt manipulert med støy, blur, endring i lysintensitet og rotasjon. For hver manipulering har det blitt gjort interklasse og intraklasse-sammenligning for hvert filter. Totalt for hvert filter har det blitt gjort 1732500 interklasse-sammenligninger og 14000 intraklasse-sammenligninger. Tilsammen for alle filtrene blir dette 6930000 interklasse-sammenligniger og 56000 intraklasse-sammenligninger. MATLAB beregner FRR og FAR direkte når sammenligningene er ferdige. I tillegg blir stolpehistogrammet med Hammingfordelingen generert automatisk.

For å se hvilke filter som er best under de forskjellige forholdene, og for å undersøke under hvilke forhold det er mulig å sammenligne iriskoder, må FRR og FAR undersøkes sammen. Ved hjelp av beregningene av FRR og FAR i MATLAB brukes disse verdiene videre i excel for å lage linjediagrammer. Det er individuelt for hvert filter hvor store Hammingdistanser som genereres, derfor vil det ved evaluering undersøkes FAR og FRR i punktet der disse skjærer hverandre. Det vil da bli undersøkt hvor stor FRR og FAR er ved nærmeste terskelverdi fra dette skjæringspunktet. Desto lavere FRR og FAR er ved dette skjæringspunktet, desto bedre matchprosent er det. Siden FAR vektlegges mer enn FRR vil den optimale terskelverdien være nærmeste punkt fra skjæringspunktet der FAR er lavere en FRR.

Det vil også bli sett på hvor mye Hammingfordelingen forandrer seg ved de forskjellige parametrene i forhold til ved optimale forhold uten forandring. Desto mer stabil denne fordelingen er under forskjellige forhold, desto lettere vil det være å bestemme en terskelverdi som kan settes ved bruk av aktuellt filter. I neste kapittel presenteres og analyseres resultatene som er oppnådd i eksperimentet.

4 Analyse av resultater

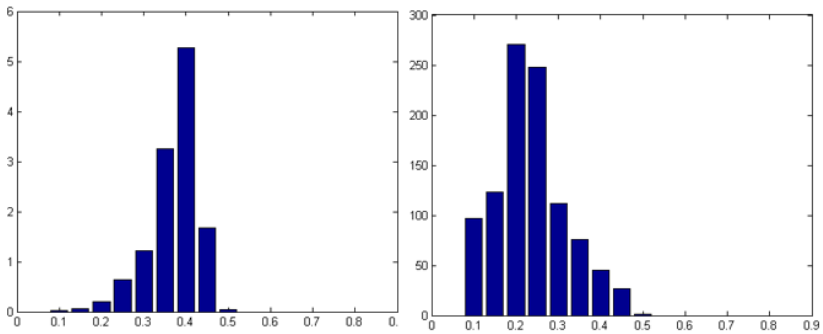
I dette kapitlet presenteres og analyseres resultatene som er oppnådd i eksperimentene. Kapitlet inneholder resultater for alle de fire implementerte filtrene. Resultater og tilhørende diskusjon presenteres fortløpende, og til slutt i hvert delkapitel gis det en oppsummering. Det vil bli fokusert på hvor mye FRR og FAR forandrer seg ettersom databasen manipuleres. For hvert filter under forskjellige forhold vil det bli foreslått en beslutningsterskel som er optimal for filteret. Figurene som viser Hammingfordelingen er med i første delkapitel, resten ligger i appendiks. Henvisninger gis underveis i teksten. Stolpehistogrammene med Hammingfordelingen viser hvordan Hammingverdiene fordeles for hvert filter. FRR og FAR beregnes ut ifra Hammingfordelingen, og disse verdiene er plottet i linjediagrammene for hvert filter for å finne den optimale beslutningsterskelen.

4.1 Analyse av resultater uten forandring i databasen

I dette delkapitlet analyseres resultatene fra eksperimentet der filterene ble testet mot databasen på 500 bilder uten noen forandringer. Filtrene analyseres hver for seg før det til slutt oppsummeres. Videre i analysen vil det bli sett på hvordan Hammingfordeligne og resultatene forandrer seg i forhold til disse resulatene, og hvilket filter som er best under de forskjellige forhldene.

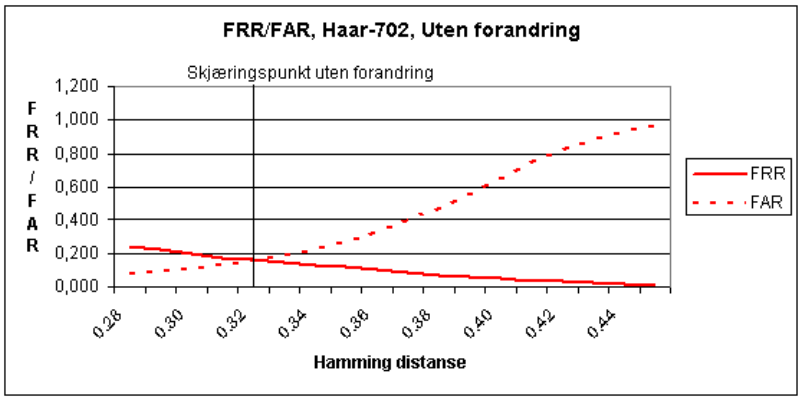
4.1.1 Analyse av 702-bit Haarfilter

Figur 18 viser Hammingfordelingen med 702-bit Haarfilter uten forandring i databasen. De fleste Hammingverdiene ved interklasse-sammenligningen ligger mellom 0.30 og 0.45. Ved intraklasse-sammenligningen genererers det forholdsvis lave Hammingverdier der de fleste verdiene ligger under 0.30.



Figur 18: Illustrasjon av Hammingfordelingen med 702-bit Haarfilter, uten forandring. Histogrammet til venstre viser interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

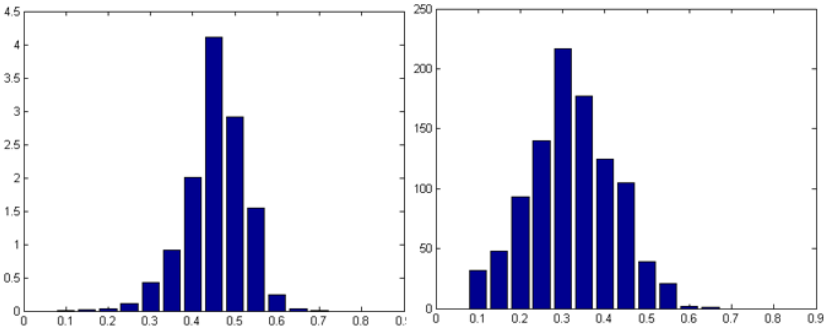
Figur 19 viser FRR og FAR ved forskjellige beslutningsterskeler. FRR og FAR skjærer hverandre i punktet der de har samme verdi, i dette tilfellet ved terskelverdi 0.33. Siden FAR vektes mer enn FRR, vil en optimal terskelverdi for 702-bit Haarfilter uten forandring være 0.32. Dette resulterer i en FRR/FAR på 0.159/0.156. Som figuren viser er avstanden mellom FRR og FAR forholdsvis liten ved lave terskelverdier, mens FAR avtar ved høyere verdier. Dette kommer av at mange Hammingverdier fra sammenligningen mellom ulike personer ligger under de høye terskelverdiene. Det motsatte ser vi ved intraklasse-sammenligningen der FRR blir lavere ved høyere terskelverdier.



Figur 19: Illustrasjon av FRR og FAR med 702-bit Haarfilter uten forandring i testdatabasen.

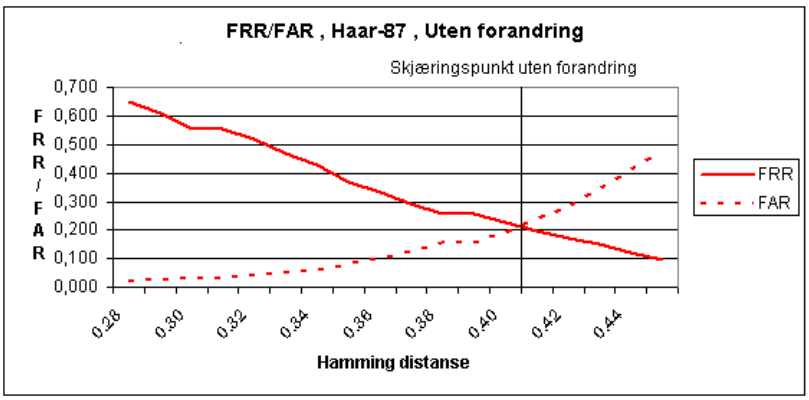
4.1.2 Analyse av 87-bit Haarfilter

Figur 20 viser Hammingfordelingen med 87-bit Haarfilter. Med dette filteret blir Hammingverdiene mellom irisvektorene større sammenlignet med filteret på 702-bit. De fleste verdiene ved interklasse-sammenligningen ligger mellom 0.35 og 0.50. Ved intraklasse-sammenligningen ligger de fleste verdiene under 0.40. I intraklasse-sammenligningen er intervallet mellom vektorene stort og fordelingen er veldig spredt til tross for at det bare er sammenligninger mellom iriser fra samme person. Dette betyr at en optimal beslutningsterskel for dette filteret vil ligge høyere enn med vektoren på 702 bit, i dette tilfellet på 0.40.



Figur 20: Hammingfordelingen med 87-bit Haarfilter, uten forandring. Histogrammet til venstre viser interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

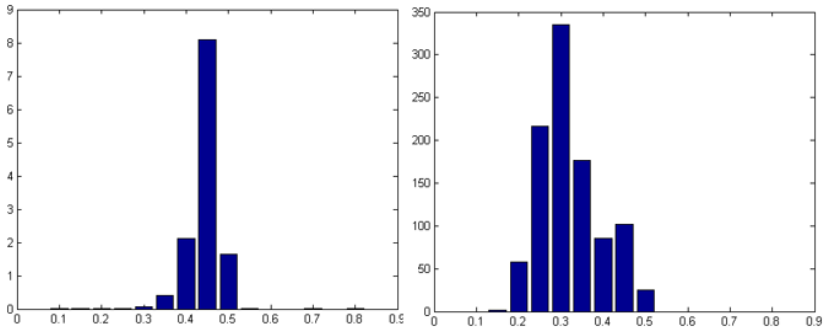
Figur 21 viser FRR og FAR med 87-bit Haarfilter. Vi ser her en tydelig forksjell mellom FAR og FRR ved lave terskelverdier. Grunnen til dette er at Hammingverdiene som genereres både ved inter og intraklasse-sammeligning er forholdsvis høye. Et resultat av dette er at FRR vil være høy ved lave beslutningsterskeler. Det motsatte ser vi med FAR som er veldig lav ved lave terskelverdier. FRR og FAR skjærer hverandre ved hd lik 0.41. Siden FAR skal vektlegges mer vil en optimal beslutningsterskel for dette filteret ved optimale forhold være 0.40. Vi oppnår i dette tilfellet en FRR/FAR på 0.226/0.189. Dette er noe dårligere resultat sammenlignet med filteret på 702-bit under samme forhold, noe som kan skyldes at dette filteret ikke klarer å filtrere ut så mange prominente detaljer fra irisen. Videre kan dette ha noe med kvaliteten på bildene å gjøre, som muligens ikke er bra nok for et høyfrekventfilter som genererer irisvektorer på kun 87-bit.



Figur 21: Illustrasjon av FRR og FAR med 87-bit Haarfilter uten forandring i testdatabasen.

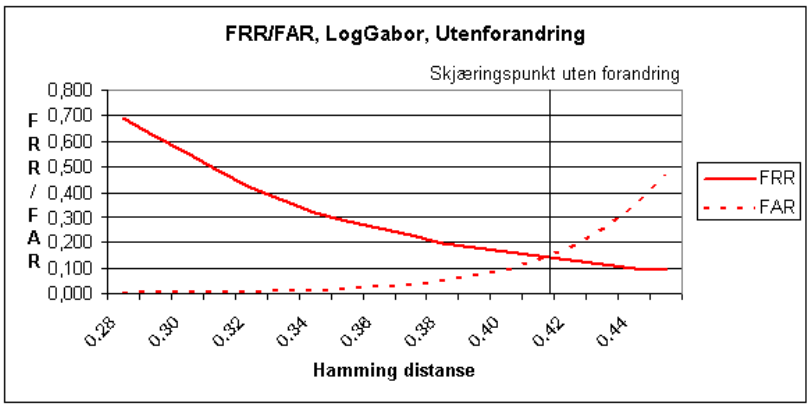
4.1.3 Analyse av Log-Gabor filter

Hammingfordelingen med Log-Gabor filteret vises i figur 22. Interklasse-fordelingen for dette filteret gir en jevnere og mer stabil fordeling sammenlignet med de foregående filterene, noe som skyldes at dette filteret henter ut mange flere tilgjenglige detaljer fra iris. De fleste av Hammingverdiene ligger rundt 0.45. Fordelingen er ikke så stabil ved intra-klasse sammenligningen der verdiene brer seg over et større intervall. De fleste Hammingverdiene her er spredt fra 0.25 til 0.45.



Figur 22: Hammingfordelingen med Log-Gabor filter, uten forandring. Histogrammet til venstre viser interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

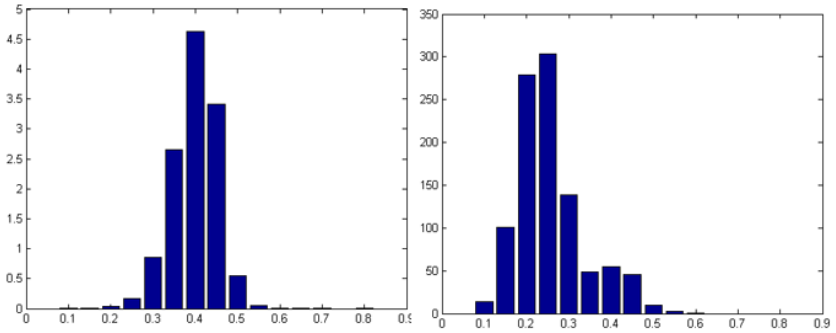
Figur 23 viser FRR og FAR med Log-Gabor filter uten forandring i databasen. Også her får vi en klar og tydelig forskjell mellom FRR og FAR ved lave terskelverdier. Vi ser at FAR er tilnærmet lik null fra beslutningsterskel 0.28-0.34. Dette kommer som sagt av den jevne og stabile interklasse-fordelingen hvor de fleste verdiene ligger rundt 0.45. FRR og FAR skjærer hverandre i underkant av 0.42. Med dette vil en beslutningsterksel på 0.41 være optimal.



Figur 23: Illustrasjon av FRR og FAR med Log-Gabor filter, uten forandring i testdatabasen.

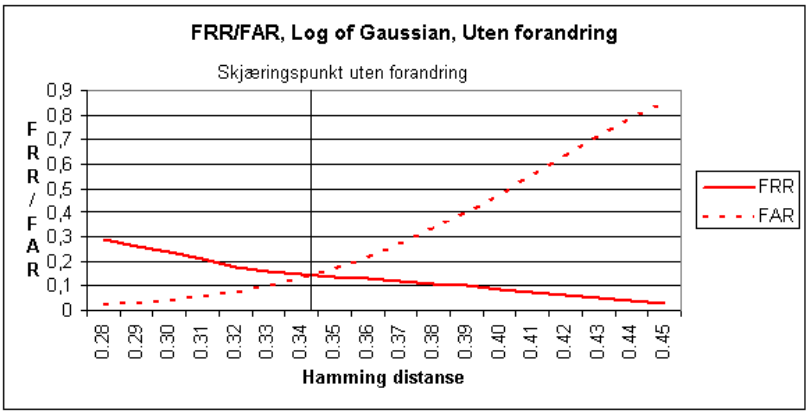
4.1.4 Analyse av Laplacian of Gaussian filter

Interklasse-sammenligningen med Laplacian of Gaussian filteret gir også en jevn fordeling der de fleste verdiene ligger mellom 0.35 og 0.45. Iriskoder som er generert med Laplacian of Gaussian filteret gir lave Hammingverdier ved Intraklasse-sammenligningen, de fleste ligger mellom 0.15 og 0.30. Fordelingen er også her noe ujevn, men verdiene er lave, noe som også gjør at den optimale terskel verdien for dette filteret blir lavere sammelignet med Log-Gabor filteret som genererer iriskoder med like mange bit. Figur 24 illustrerer Hammingfordelingen for Laplacian of Gaussian filter ved optimale forhold.



Figur 24: Hammingfordelingen med Laplacian of Gaussian filter, uten forandring. Histogrammet til venstre viser interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

Figur 25 viser FRR og FAR med Laplacian of Gaussian filteret. Dersom vi sammenligner dette filteret med Log-Gabor filteret, så er resultatet uten forandring av databasen nesten identisk. Siden Hammingverdiene som genereres er lavere både ved interklasse og intraklasse-sammenligning, vil også den optimale beslutningsterskelen være lavere, mens resultatet vil være likt. FRR og FAR skjærer hverandre i underkant av terskel verdi 0.35. En optimal terskel verdi for Laplacian of Gaussian filteret uten forandring i databasen vil da være 0.34.



Figur 25: Illustrasjon av FRR og FAR med Laplacian of Gaussian filter, uten forandring i testdatabasen.

4.1.5 Oppsummering av resultater uten forandring

Vi ser ut ifra resultatene her en liten forskjell i ytelse under optimale forhold. Det er tydelig å se at Log-Gabor og Laplacian of Gaussian filterene som genererer iriskoder med mange detaljer er best under optimale forhold. Dette kommer av detaljene i irisen som kommer tydelig frem uten andre forstyrrende elementer. I diskusjonen må det også påpekes at bildene i databasen også er med på å påvirke resultatene, da kvaliteten på disse varierer noe. Tabell 1 oppsummerer eksperimentet uten forandring av bildene i databasen. Tabellen gir også en oversikt over optimale beslutningsterskeler som gir den samlede laveste FRR og FAR prosenten, der FAR er vektlagt mer enn FRR.

	FRR/FAR	Terskel verdi
Haar-702 bit	0.159/0.156	0.32
Haar-87 bit	0.226/0.189	0.40
Log-Gabor	0.147/0.131	0.41
Laplacian of Gaussian	0.147/0.132	0.34

Tabell 1: Oversikt over ytelsen til filterene uten forandring i databasen

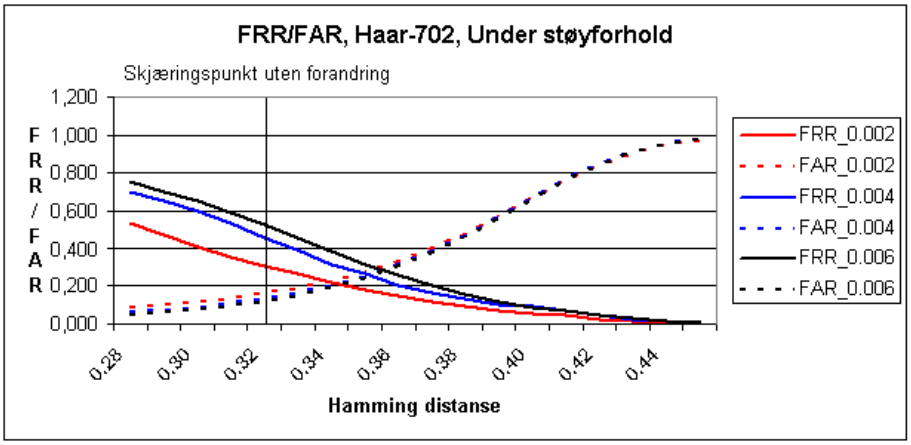
Som tabell 1 viser er Log-Gabor og Laplacian of Gaussian filterene de som gir best resultat. Resultatene er så og si helt like. Dette kan som sagt skyldes at disse filterene henter mange flere detaljer fra irisen som er tilgjenglige under disse forholdene. Log-Gabor filteret gir også en jevnere Hammingfordeling ved interklasse-sammenligning. Vi ser at den optimale terkelverdien er veldig forskjellig for disse to filterene selv om resultatene er nesten like. Vi ser også forskjell på Haarfilterene. Filteret på 702 bit gir en lavere FRR/FAR, og er på høyde med Log-Gabor og Laplacian of Gaussian til tross for den store forskjellen i størrelsene på iriskodene. Haarfilteret på 87-bit gir dårligst resultat totalt. Dette kan skyldes at bildene som er i databasen ikke er optimale nok for dette filteret som kun består av 87-bit. Resutlatene som er presentert og analysert her vil også bli brukt videre i dette kapitelet. Vi vil se hvordan resultatene endrer seg når bildene i databasen blir manipulert med forskjellige parametere, og hvilket filter som er best under de forskjellige forholdene.

4.2 Analyse av resultater med støy i databasen

I denne delen presenteres og analyseres resultatene for filterene med Gaussian støy i testdatabasen. Her blir linjediagrammene for False Acceptance Rate (FAR) og False Rejection Rate (FRR) presentert. Stolpehistogrammene som viser Hammingfordelingen ligger i appendiks A. Tabell 2 gir en oversikt over resultatene i dette eksperimentet.

4.2.1 Analyse av 702-bit Haarfilter

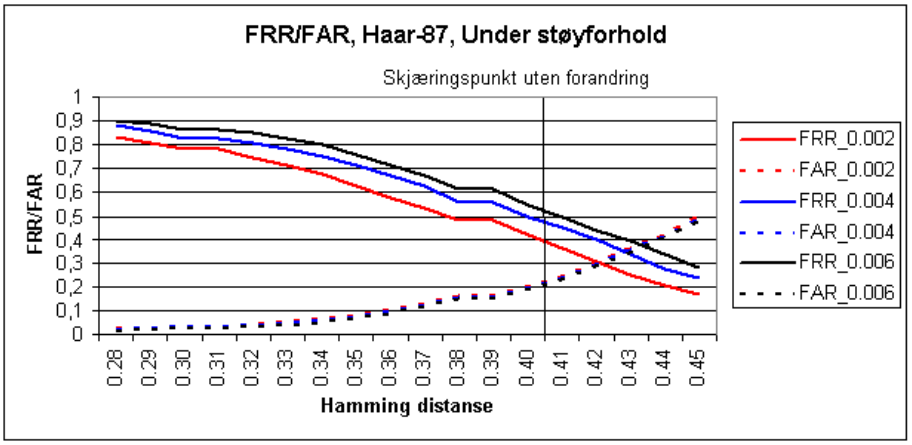
Figur 26 viser resultatet for 702-bit Haarfilter med støy i testdatabasen. Vi ser her en klar og tydelig forandring av resultatet sammenlignet med resultatene der det ikke er gjort noen forandringer. FRR og FAR skjærer hverandre ved høyere terskelverdier, og resultatet blir dårligere jo mer støy det er i bildene. En naturlig årsak til dette er at støy i bildene fjerner detaljer i irismønsteret, noe som gjør det vanskeligere å sammenligne. Som en følge av dette blir Hammingverdiene ved intraklasse-sammenligningen større når det blir lagt til støy i bildene(se også figur 46,47. og 48 i appendiks A).



Figur 26: Illustrasjon av FRR og FAR for 702-bit Haarfilter, med støy i testdatabasen.

4.2.2 Analyse av 87-bit Haarfilter

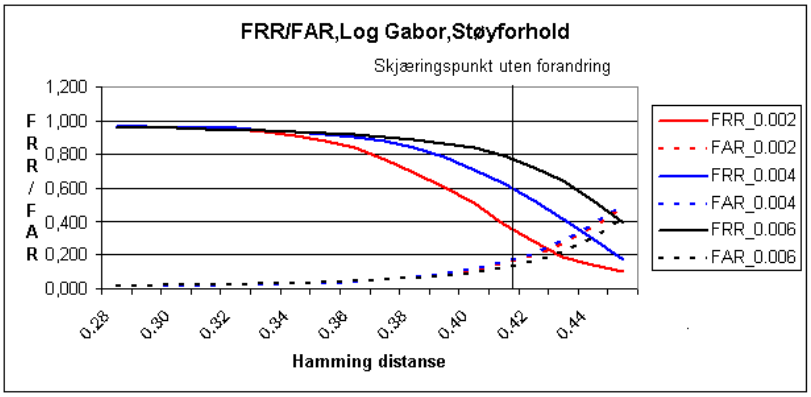
Figur 27 viser resultatet med 87-bit Haarfilter under støyforhold. Vi ser også her en klar og tydelig forskjell sammenlignet med resultatene uten forandring. Støymanipuleringen har også her størst innvirkning på intraklasse-sammenligningen og FRR, da verdiene ligger noe høyere(se også figur 49, 50 og 51 i appendiks A). Skjæringspunktet ved optimale forhold ligger også noe lavere i forhold til med støy i bildene. Vi ser en gradvis økning av FRR fra støyvarianse 0.002 til 0.006.



Figur 27: Illustrasjon av FRR og FAR med 87-bit Haarfilter, med støy i testdatabasen.

4.2.3 Analyse av Log-Gabor filter

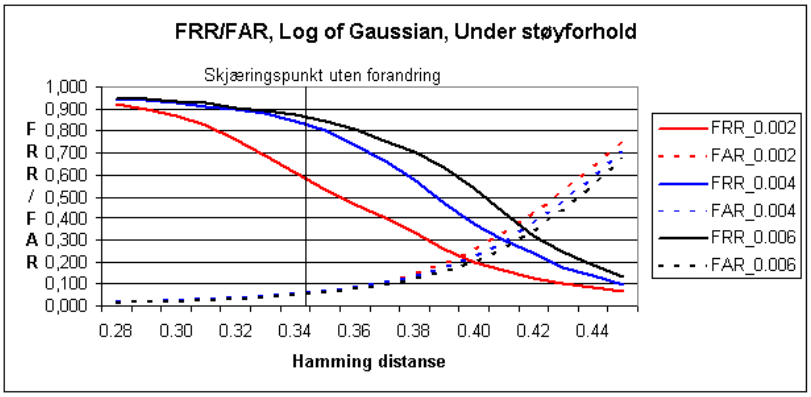
Vi ser den samme utviklingen for Log-Gabor filteret, som vises i figur 28. Linjene som representerer FRR ved de forskjellige støyvariansene ligger tett opp mot 1 før de går nedover å krysser FAR. Dette betyr at nesten alle Hammingverdiene mellom iriser av samme person har en Hamming verdi over 0.34(se også figur 52, 53 og 54 i appendiks A). Som linjediagrammet viser blir resultatet dårligere jo mer støy som legges til i bildene. Dette gjelder i størst grad resultatene ved intraklasse-sammenlingning.



Figur 28: Illustrasjon av FRR og FAR for Log-Gabor filter, med støy i testdatabasen.

4.2.4 Analyse av Laplacian of Gaussian filter

Også med Laplacian of Gaussian filter forandrer resultatene seg veldig mye ved intraklasse-sammenligning. Figur 29 viser dette(se også figur 55, 56 og 57). Vi ser at FRR ligger veldig høyt ved lave beslutningsterskeler. Vi ser at interklasse-sammenligningen forandrer seg lite, noe som indikerer at Hammingverdiene blir noe større med støy i bildene. Vi ser også at FAR ved forskjellig støynivå har en slakere stigningskurve sammenlignet med resultatene da det ikke ble gjort forandringer i databasen, noe som kan skyldes høyere Hammingverdier ved interklasse-sammenligning.



Figur 29: Illustrasjon av FRR og FAR med Laplacian of Gaussian filter, med støy i testdatabasen.

4.2.5 Oppsummering av resultater med støy i databasen

Tabell 2 viser oversikten over resultatene i eksperimentet under støyforhold. F1 er 702-bit Haarfilter, F2 er 87-bit Haarfilter, F3 er Log-Gabor filter og F4 er Laplacian of Gaussian filter. Som vi ser så blir altså resultatene dårligere for alle filterene, og jo mer støy som legges til i bildene jo dårligere blir resultatene. Grunnen til dette er som nevnt tidligere at støy gjør det vanskeligere å filtrere ut og kode detaljene i irismønsteret. Dette gjør at avstanden mellom iriser fra samme person blir større fordi det blir færre like detaljer fra irisen som kan sammenlignes, samtidig som avsanden mellom iriser fra ulike personer ikke påvirkes i samme grad. Tabellen viser at 702-bit Haarfilter er mest robust under støyforhold selv om variasjonene er små. Dette kan skyldes at 702-bit Haarfilter filtrerer ut irisdetaljene ved hjelp av færre bit (sammenlignet med Log-Gabor og Laplacian of Gaussian), men henter de mest tydelige og særegne detaljene i irisen, og blir ikke så forstyrret av støy i bildet. Vi ser også at 87-bit Haarfilter ikke er så robust mot støy. Dette filteret genererer irisvektorer basert på høypass filterering, noe som tydeligvis har stor innvirkning når det legges til støy i bildene og det blir færre detaljer i irisen å hente ut.

	Støyvarianse 0.002			Støyvarianse 0.004			Støyvarianse 0.006		
	Terskel	FRR	FAR	Terskel	FRR	FAR	Terskel	FRR	FAR
F1	0.33	0.265	0.189	0.35	0.262	0.249	0.36	0.315	0.302
F2	0.42	0.304	0.298	0.42	0.397	0.291	0.43	0.392	0.344
F3	0.42	0.280	0.196	0.43	0.412	0.277	0.44	0.528	0.295
F4	0.39	0.259	0.191	0.41	0.300	0.287	0.42	0.384	0.336

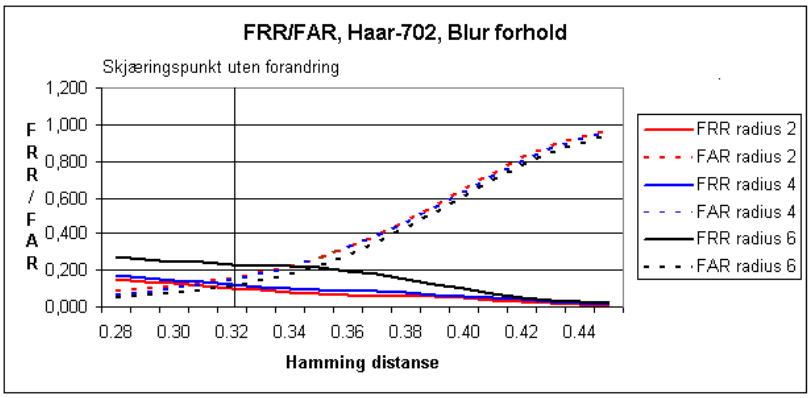
Tabell 2: Oversikt over ytelsen til filterene med støy i databasen

4.3 Analyse av resultater med uskarphet

I denne delen presenteres resultatene for filterene med uskarphet (blur) i testdatabasen. Her blir linjediagrammene for FRR og FAR for hvert filter presentert. Stolpehistogrammene som viser Hammingfordelingen ligger i appendiks B. Tabell 3 gir en oversikt over resultatene i dette eksperimentet.

4.3.1 Analyse av 702-bit Haarfilter

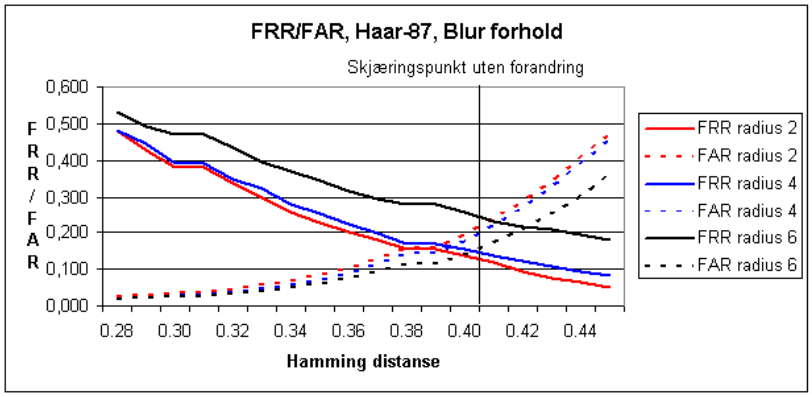
Figur 30 viser resulatene med 702-bit Haarfilter der bildedatabasen er manipulert med forskjellig nivå med uskarphet. Den største forskjellen sammenlignet med resultatene uten forandring får vi ved intraklasse-sammenligningen. Vi får her lavere Hammingverdier ved blur radius 2 og 4 (se også figur 58, 59 og 60 i appendiks B). Dette kan skyldes at når bildene blir lagt til små mengder blur, så lavpassfiltreres bildene slik at detaljene i irisen kommer tydeligere frem. Med flere tilgjenglige detaljer å sammenligne jo lettere er det å finne likheter og ulikheter mellom irisene, dermed blir også resultatene bedre. Med radius seks oppnår vi dårligere resultater fordi her blir uskarpheten så stor i bildene at detaljene blir mer borte i irisen.



Figur 30: Illustrasjon av FRR og FAR for 702-bit Haarfilter, med blur i testdatabasen.

4.3.2 Analyse av 87-bit Haarfilter

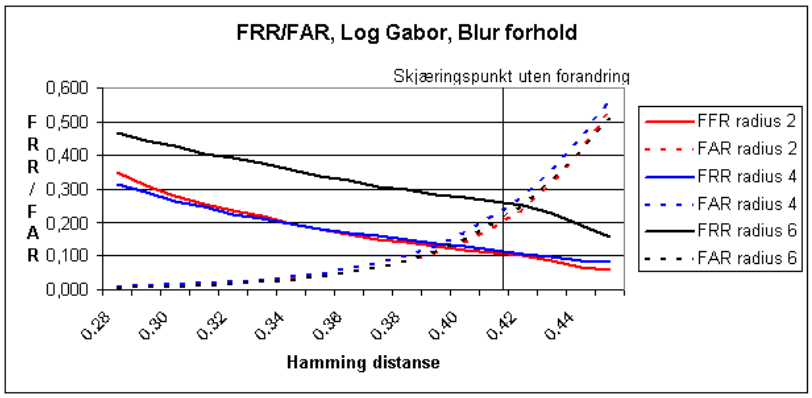
Vi ser samme utvikling for Haarfilter med 87-bit, figur 31 viser dette. Hammingveridene ved intraklasse-sammenligningen er noe lavere med blur radius 2 og 4, mens ved interklasse-sammenligningen blir de noe høyere. Med radius 6 blir resultatene dårligere, noe som også var tilfellet med 702-bit Haar filter. Dersom vi ser på bildene med blur i figur 15, som er illustrert i eksperiment oppsettet, ser vi en klar og tydelig forskjell fra bildet med radius 4 til bildet med radius 6. Vi ser at det blir færre detaljer i irisen noe som også resultatene bærer preg av(se også figur 61, 62 og 63 i appendiks B). Den tydeligste forskjellen ser vi ved intraklasse-sammenligningen og FRR.



Figur 31: Illustrasjon av FRR og FAR for 87-bit Haarfilter, med blur i testdatabasen.

4.3.3 Analyse av Log-Gabor filter

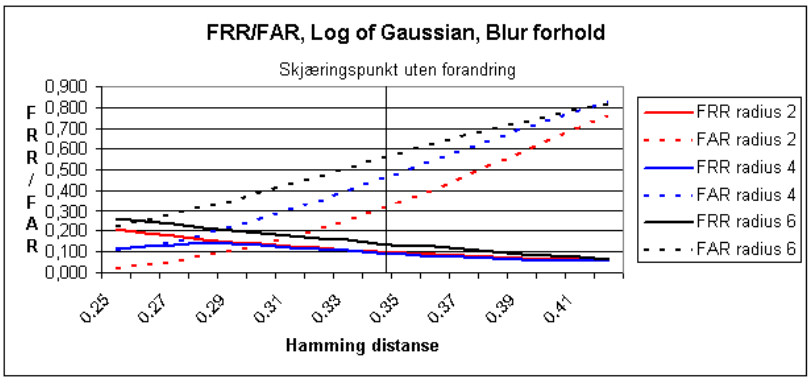
Med Log-Gabor filteret ser vi samme tendensen som med de foregående filtrene, se figur 32. Vi ser her at med blur radius 2 og 4 så skjærer FRR og FAR hverandre under det optimale punktet. Vi ser at Hammingfordelingen ved intraklasse-fordelingen ligger en del lavere sammenlignet med resultatene uten forandring (se også figur 64, 65 og 66 i appendiks B). Grunnen til dette er som nevnt tidligere at bildene blir lavpassfiltrerte når det blir lagt til små mengder blur, som resulterer i at flere teksturer i irisen kan sammenlignes. Vi ser en klar og tydelig forskjell med blur radius 6 for dette filteret, da resultatene blir veldig mye dårligere.



Figur 32: Illustrasjon av FRR og FAR for Log-Gabor filter, med blur i testdatabasen.

4.3.4 Analyse av Laplacian of Gaussian filter

Vi ser ut ifra figur 33 at Hammingfordelingen ved intraklasse-sammenlignigen også her gir lavere Hammingverdier med radius 2 og 4 (se også figur 67, 68 og 69 i appendiks B). Vi ser også at en del av Hammingverdiene ved interklasse-sammenligningen blir noe større noe som fører til lavere FAR. Totalt sett så blir også resultatet her bedre med blur radius to og fire i testdatabasen. En del detaljer i irisen blir borte når bildet inneholder mye blur, derfor blir også resultatet dårligere med blur radius 6 i databasen.



Figur 33: Illustrasjon av FRR og FAR for Laplacian of Gaussian filter, med blur i testdatabasen.

4.3.5 Oppsummering av resultater med uskarphet i databasen

Tabell 3 viser oversikten over resultatene i eksperimentet under uskarphetsforhold. F1 er 702-bit Haarfilter, F2 er 87-bit Haarfilter, F3 er Log-Gabor filter og F4 er Laplacian of Gaussian filter. Vi ser den samme tendensen hos samtlige filtere der resultatene blir bedre når det blir lagt til litt blur i bildedatabasen. Som nevnt tidligere kan dette skyldes at bildene i databasen lavpassfiltreres, slik at detaljene i irisen fremheves. Da blir avstanden mellom iriser fra samme person mindre fordi det er flere like detaljer fra irisen som kan sammenlignes. Det motsatte skjer når det sammenlignes iriser fra ulike personer. Jo flere delaljer fra iris som filtreres ut jo større kan avstanden bli mellom ulike personer. Dette er da tilfellet når det blir lagt til blur med radius 2 og 4, mens med radius 6 ser vi det motsatte. Med blur radius 6 (se figur 15) blir irisen så uskarp at detaljene blir mindre fremhevet.

	Blur radius 2			Blur radius 4			Blur radius 6		
	Terskel	FRR	FAR	Terskel	FRR	FAR	Terskel	FRR	FAR
F1	0.30	0.126	0.118	0.31	0.134	0.122	0.34	0.221	0.180
F2	0.37	0.181	0.128	0.40	0.178	0.154	0.43	0.216	0.210
F3	0.39	0.127	0.109	0.39	0.138	0.128	0.41	0.263	0.195
F4	0.30	0.137	0.135	0.27	0.155	0.148	0.25	0.251	0.223

Tabell 3: Oversikt over ytelsen til filterene med uskarphet i databasen

Som tabell 3 viser ser vi at det er med 702-bit Haarfilter vi totalt oppnår best resultat med under disse forholdene, selv om variasjonene er forholdsvis små. Med dette filteret oppnådde vi best resultat under de største endringene (radius 4 og 6). Nok en gang kan dette skyldes at 702-bit Haarfilter filtrerer ut iris detaljene ved hjelp av færre bit, men

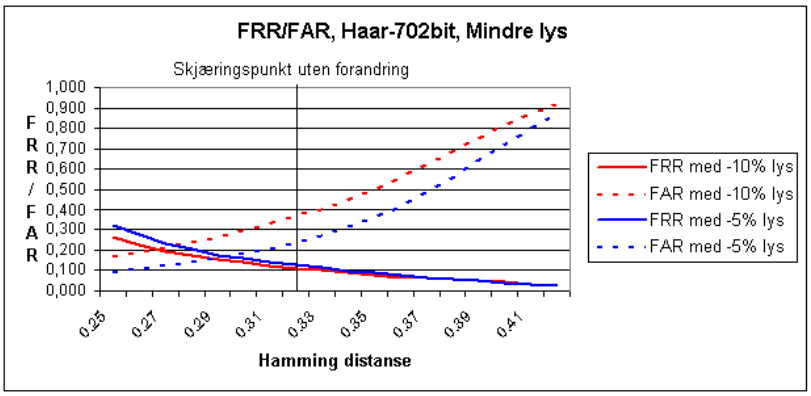
henter de mest tydelige og fremhevede detaljene i irisen, og blir ikke så forstyrret av de største endringene. Vi ser også at det er med Log-Gabor filter vi oppnår best resultat med når det blir lagt til blur radius 2.

4.4 Analyse av resultater med lysforandring

I denne delen presenteres resultatene for filterene med forandring av lysintensiteten i testdatabasen. Her blir linjediagrammene for FRR og FAR ved hvert filter presentert. Stolpehistogrammene som viser Hammingfordelingen ligger i appendiks C.

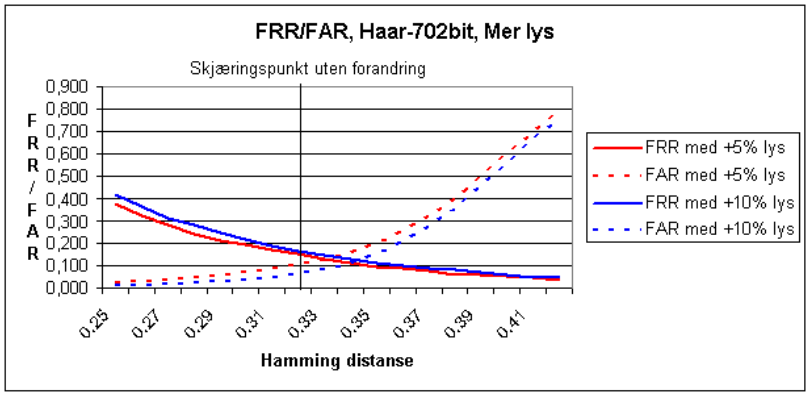
4.4.1 Analyse av 702-bit Haarfilter

Figur 34 viser resultatene med 702-bit Haarfilter der lyset i bildedatabasen er redusert med 5 og 10 prosent. Vi ser at FRR og FAR skjærer hverandre ved lavere beslutnigsterskeler sammenlignet med resultatene der det ikke ble gjort forandringer. Dette skyldes at når lyset i bildene reduseres, reduseres også detaljene i irisen, og dette gjør det vanskeligere å filtrere ut og kode irismønsteret. Hammingverdiene ved interklasse-sammenligning blir mindre, og FAR vil her ligge høyere. Dette gjelder særlig med 10 prosent mindre lys(se også figur 70 og 71 i appendiks C). Det motsatte ser vi ved intraklasse-sammenligning der Hammingveridene blir noe større. Dette fører til at FRR vil ligge noe høyere.



Figur 34: Illustrasjon av FRR og FAR for 702-bit Haarfilter, med 10 og 5 prosent reduksjon av lysintensiteten i testdatabasen.

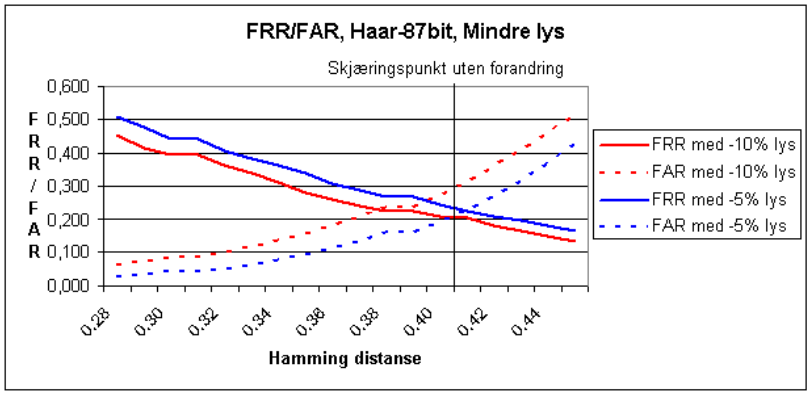
Når vi øker lysintensiteten i testdatabasen med 5 og 10 prosent, ser vi av figur 35 at resultatene blir bedre. Med mer lys i bildene gjør at det blir flere synlige og gjenkjennlige detaljer/strukturer i irisen som kan filtereres ut og kodes. Dette gjør det lettere i sammenligningen å finne større avstand mellom iriser fra forskjellige personer(interklasse-sammenligning) og mindre mellom iriser fra samme person(intraklasse-sammenligning). Se også figur 72 og 73.



Figur 35: Illustrasjon av FRR og FAR for 702-bit Haarfilter, med 5 og 10 prosent økning av lysintensiteten i testdatabasen.

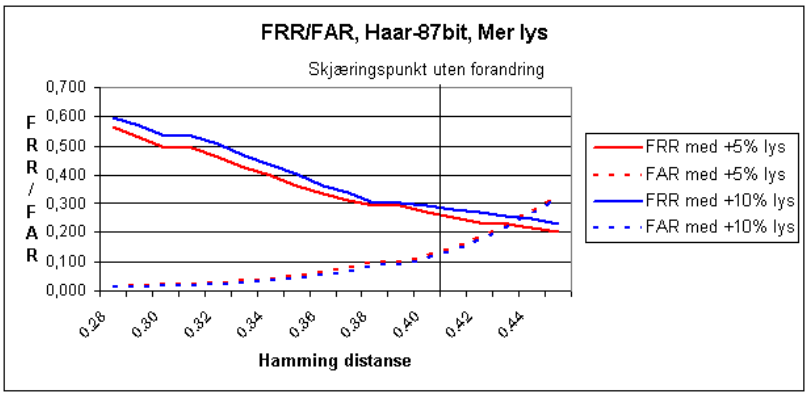
4.4.2 Analyse av 87-bit Haarfilter

Figur 36 viser resultatet for 87-bit Haarfilter der lysintensintensiteten er redusert med 5 og 10 prosent. Med en reduksjon på 10 prosent ser vi at FRR og FAR skjærer hverandre under den optimale beslutningsterskelen som ble satt uten forandring i databasen. Vi ser også at med 10 prosent reduksjon i lysintensiteten blir Hammingverdiene ved interklassesammenligningen noe mindre(se også figur 74 og 75 i appendiks C). Det er liten forskjell i resultatet på ti og fem prosent reduksjon i lysintensitet med dette filteret.



Figur 36: Illustrasjon av FRR og FAR for 87-bit Haarfilter, med 10 og 5 prosent reduksjon av lysintensiteten i testdatabasen.

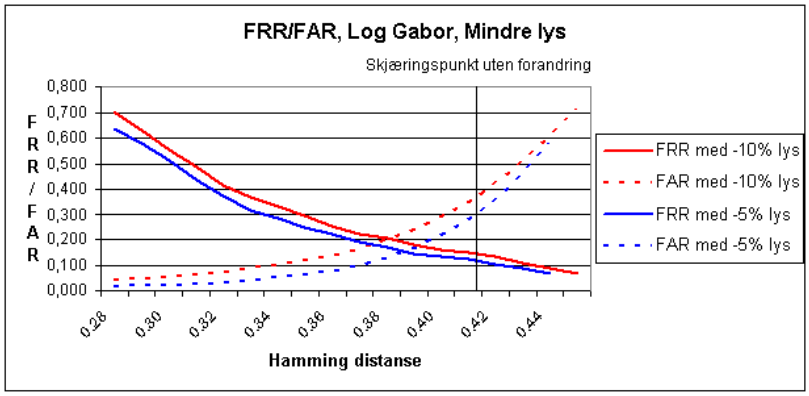
Som vi ser av figur 37 er FAR både ved 5 og 10 prosent økning nesten helt like. Vi ser at under disse forholdene vil en optimal terskelverdier for dette filteret være høyere sammenlignet med resultatet uten forandring. Dersom vi ser på Hammingfordelingen(figur 76 og 77 i appendiks C) ser vi at en del av Hammingverdiene blir lavere ved intraklasse-sammenligning. Ved interklasse sammenligning er ikke denne forskjellen så stor. Det ser ikke ut til at resutlatene har forandret seg mye med økning i lysintensitet. Grunnen til dette kan nok skyldes at vektoren på 87-bit er for liten til å dra nytte av mer lys i bildene, og å representere nok informasjon om irisen under disse forholdene.



Figur 37: Illustrasjon av FRR og FAR for 87-bit Haarfilter, med fem og ti prosent økning av lysintensiteten i testdatabasen.

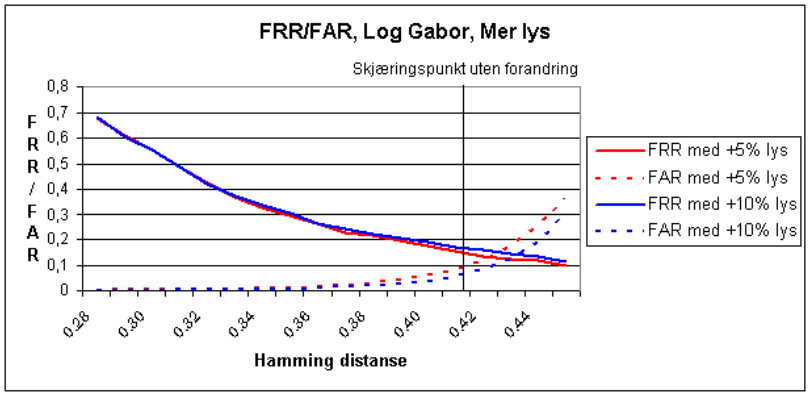
4.4.3 Analyse av Log-Gabor filter

Figur 38 viser resultatene med Log-Gabor filter der lysintensiteten i bildedatabasen er redusert med 5 og 10 prosent. Vi ser at med 10 prosent reduksjon så ligger FRR en del høyere sammenlignet med da filteret ble kjørt på databasen uten forandringer. Det samme gjelder for FAR, da vi ser at denne stiger raskere. Også her blir Hammingverdiene ved interklasse-sammenligningen lavere, mens ved intraklasse blir verdiene noe høyere(se også figur 78 og 79 i appendiks C). Med 5 prosent reduksjon i lysintensitet oppnår vi et forholdsvis uforandret resultat i forhold til da det ble eksperimentert med en database uten forandring. Her er det lite som skiller, noe som kan tyde på at dette filteret er robust mot små endringer i lysintensitet.



Figur 38: Illustrasjon av FRR og FAR for Log-Gabor filter, med ti og fem prosent reduksjon av lysintensiteten i testdatabasen.

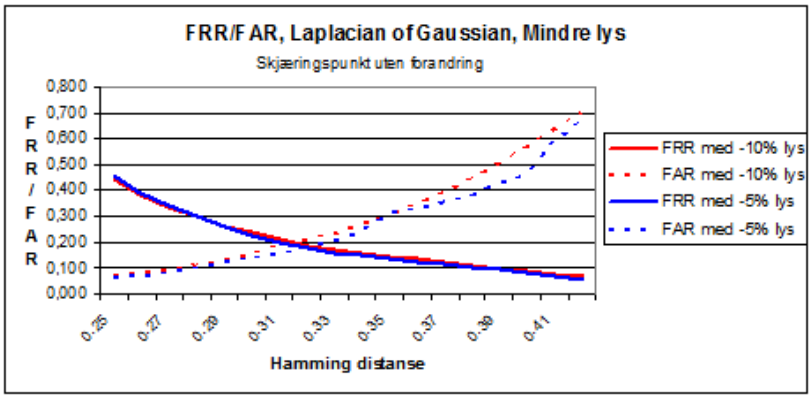
Med økning av lysintensitet ser vi samme tendens som vi gjorde med 702-bit Haarfilter(se figur 39). Vi ser også her at ved interklasse-sammenligning ligger FAR veldig lavt lenge. Grunnen til dette er som nevnt tidligere at når to forskjellige iriskoder sammenlignes med flere filtrerte detaljer, vil også Hammingverdiene mellom disse bli større (se også figur 80 og 81 i appendiks C). Da får vi mange flere forskjellige teksturer i irisen å sammneligne, noe som gjør det lettere å differensiere iriser fra forskjellige personer. Dette ser vi også av kurvene i figur 39.



Figur 39: Illustrasjon av FRR og FAR for Log-Gabor filter, med 5 og 10 prosent økning av lysintensiteten i testdatabasen.

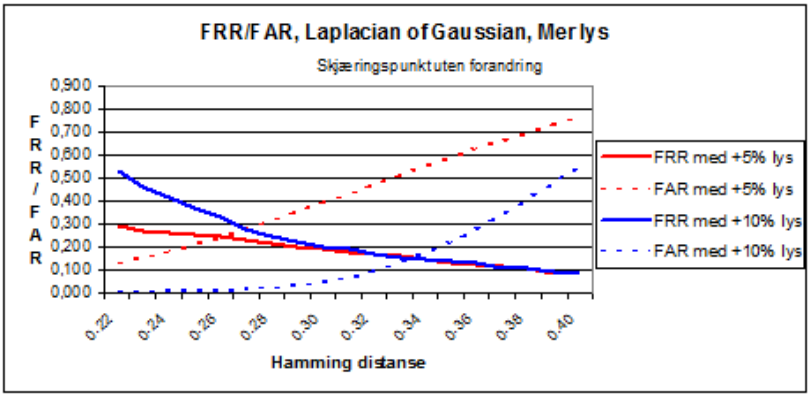
4.4.4 Analyse av Laplacian of Gaussian filter

Figur 40 viser resultatet med Laplacian of Gaussian filter med en reduksjon i lysintensiteten på 5 og 10 prosent. Vi ser ved hjelp av kurvene at resultatene er ganske like ved 10 og 5 prosent reduksjon, og at FRR og FAR fordeler seg forholdsvis likt. Grunnen til dette er også her Hammingverdiene ved interklasse-sammenlignig som ligger noe lavere sammenlignet med resultatet uten forandring(se også figur 82 og 83 i appendiks C). Vi ser det motsatte med Hammingverdiene ved intraklasse-sammenligning, da disse blir større. Denne utviklingen har vi også sett ved de andre filterene. Som resultatet viser er også dette filteret robust mot små endringer i lysintesiteten i bildene.



Figur 40: Illustrasjon av FRR og FAR for Laplacian of Gaussian filter, med 10 og 5 prosent reduksjon av lysintensiteten i testdatabasen.

Figur 41 viser resultatet for Laplacian of Gaussian filter da lysintensiteten i bildedatabasen er økt med 5 og 10 prosent. Optimale beslutningsterkeler for disse endringene varierer veldig da vi ser at det er forholdsvis stor forskjell på FAR med 10 prosent økning i forhold til 5 prosent. Resultatet med 5 prosent blir veldig mye dårligere sammenlignet med Log-Gabor og 702-bit Haarfilter. Dersom vi ser på Hammingfordelingen ved disse forholdene(figur 84 og 85 i appendiks C), ser vi at også her blir flere av Hammingverdiene ved interklasse-sammenligningen noe høyere sammenlignet med interklase-sammenligningen uten forandring. Vi får også flere lave verdier ved intraklasse-sammenligningen.



Figur 41: Illustrasjon av FRR og FAR for Laplacian of Gaussian filter, med 5 og 10 prosent økning av lysintensiteten i testdatabasen.

4.4.5 Oppsummering med forandring av lysintensiteten i databasen

Tabell 4 og 5 viser oversikten over resultatene i eksperimentet der lysintensiteten i bilde-databasen er endret. F1 er 702-bit Haarfilter, F2 er 87-bit Haarfilter, F3 er Log-Gabor filter og F4 er Laplacian of Gaussian filter. Som nevnt tidligere blir irismønsteret mer fremhevet og synlig når lysintensiteten i bildene økes. Vi ser at 702-bit Haarfilter er best under de største endringene ved henholdsvis 10 prosent reduksjon og økning. Med 5 prosent økning og reduksjon er det Log-Gabor filteret som er best. At 702-bit Haarfilter er best når lysintensiteten i bildene reduseres med 10 prosent kan nok engang skyldes at dette filteret henter ut og koder færre men mer prominente og synlige detaljer fra irisen, noe som gjør dette filteret robust mot lysendringer. Vi ser at ved 10 prosent reduksjon er det veldig lite som skiller Log-Gabor, Laplacian of Gaussian og 702-bit Haarfilter, noe som gjør disse filterene robuste mot endringer i lysintensitet.

	Lysintensitet -10%			Lysintensitet -5%		
	Terskel	FRR	FAR	Terskel	FRR	FAR
F1	0.26	0.225	0.188	0.29	0.175	0.159
F2	0.39	0.243	0.207	0.41	0.229	0.228
F3	0.38	0.205	0.198	0.39	0.168	0.127
F4	0.31	0.212	0.192	0.31	0.198	0.156

Tabell 4: Oversikt over ytelsen til filterene med mindre lys i databasen

	Lysintensitet +5%			Lysintensitet +10%		
	Terskel	FRR	FAR	Terskel	FRR	FAR
F1	0.32	0.129	0.128	0.33	0.130	0.113
F2	0.43	0.230	0.224	0.44	0.256	0.217
F3	0.42	0.135	0.120	0.43	0.146	0.122
F4	0.26	0.251	0.210	0.33	0.153	0.126

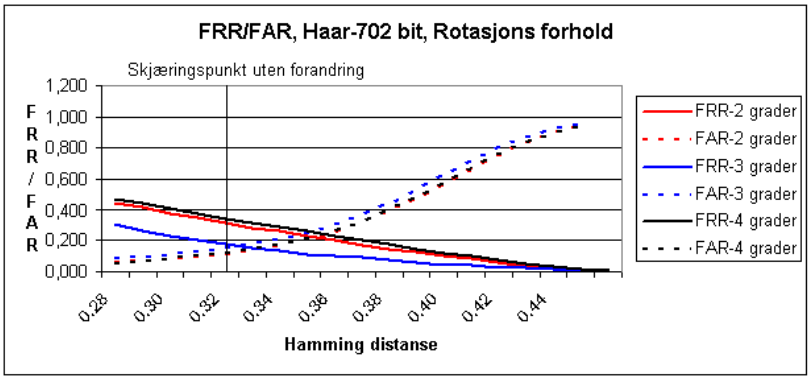
Tabell 5: Oversikt over ytelsen til filterene med mer lys i databasen

4.5 Analyse av resultater med rotasjon

I denne delen presenteres resultatene for filtrene med rotasjon av bildene i testdatabasen. Her blir linjediagrammene for FRR og FAR ved hvert filter presentert. Stolpehistogrammene som viser Hammingfordelingen ligger i appendiks D.

4.5.1 Analyse av 702-bit Haarfilter

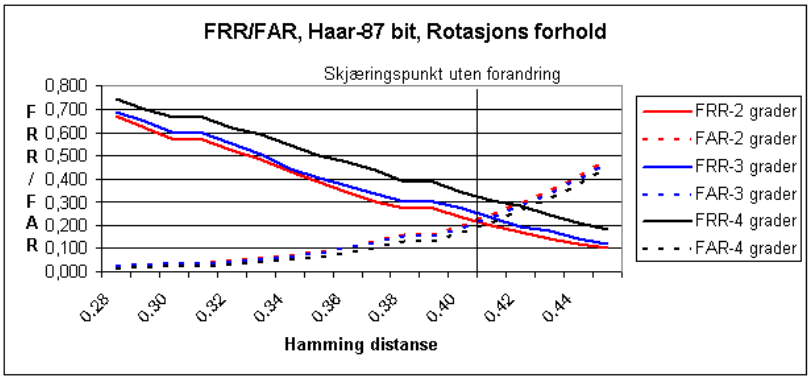
Figur 42 viser resultatene med 702-bit Haarfilter der databasen er rotert med 2, 3 og 4 grader. Vi ser at resultatene med 2 og 4 graders rotasjon blir noe dårligere sammenlignet med da det ikke ble gjort forandringer, mens med 3 grader får vi nesten uforandret resultat (se tabell 6). Dersom vi ser på Hammingfordelingen ved interklasse-sammenligningen for dette filteret(figur 86, 87 og 88 i appendiks D) så ser vi at fordelingen er veldig uforandret. Vi får en liten forandring ved intraklasse-fordelingen som gjør at FRR blir noe høyere. Dette gjelder i stor grad med 2 og 4 graders rotasjon, da en del av Hammingverdiene blir noe høyere.



Figur 42: Illustrasjon av FRR og FAR for 702-bit Haarfilter, med rotasjon av bildene i testdatabasen med 2, 3 og 4 grader

4.5.2 Analyse av 87-bit Haarfilter

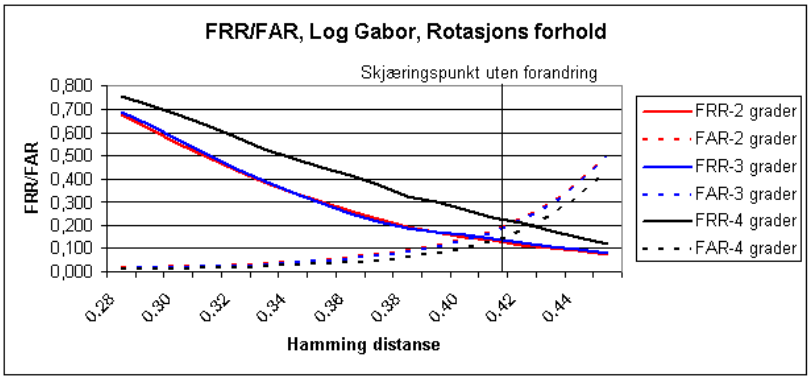
Figur 43 viser resultatet med 87-bit Haarfilter der bildedatabasen er rotert med 2, 3 og 4 grader. I dette tilfellet får vi et ganske så uforandret resultat med 2 og 3 graders rotasjon (se også tabell 6). Dersom vi ser på Hammingfordelingen (figur 89, 90 og 91 i appendiks D) så ser vi at Hammingverdiene ved intraklasse-sammenligningen gir en del høyere verdier ved 4 graders rotasjon av databasen. Dette resultater i en høyere optimal beslutningsterskel, mens med 2 og 3 graders rotasjon blir denne uforandret i forhold til terskelen som ble satt for filteret uten forandring av databasen. Interklasse-sammenligningen gir en forholdsvis uforandret Hammingfordeling. Dette tyder på at når bildene blir rotert så har dette størst innvirkning på FRR og intraklasse-sammenligningen for dette filteret.



Figur 43: Illustrasjon av FRR og FAR for 87-bit Haarfilter, med rotasjon av bildene i testdatabasen med 2, 3 og 4 grader

4.5.3 Analyse av Log-Gabor filter

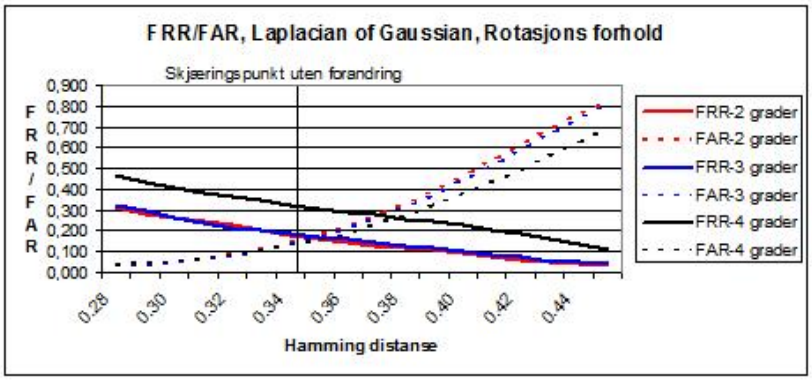
Figur 44 viser resultatet med Log-Gabor filter der bildedatabasen er rotert med 2, 3 og 4 grader. Vi ser at FAR forandrer seg lite når databasen roteres. Grunnen til dette er at Hammingverdiene ved interklasse-sammenligningen ikke forandrer ser noe særlig (se også figur 92, 93 og 94 i appendiks D). Som vi ser forandrer FRR seg veldig med en rotasjon på 4 grader. Her blir en del av Hammingverdiene ved intraklasse-sammenligningen en del større, noe som også gjør at resultatet for FAR og FRR ved en optimal beslutningsterskel vil være dårligere sammenlignet med resultatet uten forandring. FRR forandrer seg lite ved rotasjon på 2 og 3 grader, noe som gjør at resultatet for FAR of FRR her blir lite forandret i forhold til resultatene uten forandring.



Figur 44: Illustrasjon av FRR og FAR for Log-Gabor filter, med rotasjon av bildene i testdatabasen med 2, 3 og 4 grader

4.5.4 Analyse av Laplacian of Gaussian filter

Figur 45 viser resultatet med Laplacian of Gaussian filter der bildedatabasen er rotert med 2, 3 og 4 grader. Vi ser også her at rotasjon av databasen med 2 og 3 grader forandrer resultatet for FRR lite (se også 95, 96 og 97 i appendiks D). Med 4 graders rotasjon av databasen kan vi se av Hammingfordelingen at flere av Hammingverdiene ved intraklasse-sammenligningen blir høyere sammenlignet med resultatet uten forandring. Vi ser også at Hammingfordelingen for interklasse-sammenligningen ved 4 graders rotasjon gir noe høyere Hammingverdier. Med 4 graders rotasjon i seg selv vil vi få en lavere FAR, men siden FRR er såpass høy, vil det samlede resultatet for FRR og FAR ved en optimal beslutningsterskel bli dårligere sammenlignet med resultatet uten forandring.



Figur 45: Illustrasjon av FRR og FAR for Laplacian of Gaussian filter, med rotasjon av bildene i testdatabasen med 2, 3 og 4 grader

4.5.5 Oppsummering av resultater med rotasjon av databasen

Tabell 6 viser en oppsumnmering av resultatene med rotasjon av databasen. F1 er 702-bit Haarfilter, F2 er 87-bit Haar filter, F3 er Log-Gabor filter og F4 er Laplacian of Gaussian filter. Som vi ser er det med Log-Gabor filteret vi oppnår best resultater under disse forholdene. Vi ser at når databasen er rotert får vi en del dårligere resultater, noe som kan skyldes uoverensstemmelse som oppstår i normaliseringen, noe som får innvirkning på resultatene selv om det ikke er gjort noen forandringer i selve bildene. Disse uoverensstemmelsene ser vi har størt innvirkning på Haarfilterene som genererer iriskoder med langt færre detaljer fra iris, noe som gir dårligere utslag ved sammenligning.

	2 grader			3 grader			4 grader		
	Terskel	FRR	FAR	Terskel	FRR	FAR	Terskel	FRR	FAR
F1	0.35	0.230	0.209	0.32	0.179	0.151	0.35	0.229	0.213
F2	0.40	0.195	0.230	0.40	0.269	0.187	0.42	0.281	0.259
F3	0.40	0.148	0.139	0.40	0.160	0.135	0.42	0.206	0.174
F4	0.34	0.178	0.147	0.35	0.173	0.172	0.37	0.277	0.232

Tabell 6: Oversikt over ytelsen til filterene med rotasjon av databasen

5 Konklusjon

Følgende forskningsspørsmål ble definert for denne oppgaven:

- Under hvilke forhold er irisgjenkjenning mulig?
- Hvilken filteralgoritme er best under ugunstige forhold?

I eksperimentet har vi funnet ut at forholdene er veldig viktig under irisgjenkjenning, og det har vist seg at dersom bildene inneholder for mye forstyrrelser, får vi problemer med segmenteringen. Algoritmene til Masek [3] er veldig bra under optimale forhold, men viste seg å ikke være like bra under forskjellige ugunstige forhold. Det største problemet får vi altså under segmenteringsprosessen, og det er her vi ser under hvilke forhold irisgjenkjenning er mulig. Med utgangspunkt i algoritmene til Masek og med bakgrunn i eksperimentet som er utført, kan vi konkludere med følgende; Med Gaussian støy i bildedatabasen er irisgjenkjenning mulig hvis støyvariansen ikke overstiger 0.006. Dersom støyvariansen overstiger dette nivået får vi feil i segmenteringen på mange av bildene. Dette kan skyldes den lave kontrasten mellom pupill og iris, noe som fører til at det blir vanskeligere å finne sirkler i øyet og skille irisen fra pupillen. Det samme problemet fikk vi også når det ble lagt til blur i databasen. Med en blur radius på mer en 6 fikk vi feil i segmenteringen på mange av bildene, så bilder bør ikke inneholde mer enn blur radius 6. Vi fant også ut at når det blir lagt til små mengder blur i bildene, så lavpassfiltreres bildene og detaljene i irisen kommer tydeligere frem, noe som også gjør resultatene bedre ved sammenligning. Ved forandring av lysintensiteten i bildene så er pluss minus 10 prosent grensen for å kunne sammenligne to iriskoder. Noe særlig mer enn dette resulterte i at både FAR og FRR lå tett opp mot 100 prosent. Dette kan skyldes at når lyset i bildene reduseres for mye, reduseres også de tilgjenglige detaljene i irisen, noe som fører til at det ikke er nok detaljer i irisene til å kunne sammenligne to iriskoder. Når lysintensiteten i bildene økes blir detaljene i iris mer fremhevet og resultatet blir bedre. Ved rotasjon av bildedatabasen fant vi ut at med mer enn fire graders rotasjon får vi segmenteringsfeil i mange av bildene. Resultatene ved rotasjon bærer også preg av uoverenstemmelse i normaliseringen.

Med bagrunn i resultatene fra eksperimentet, kan vi videre konkludere med at ytelsen til alle algoritmene som har inngått i eksperimenetet blir veldig berørt av de nedbrytende forholdene. Vi kan konkludere med at Log-Gabor og Laplacian of Gaussian filterene som genererer iriskoder på 9600-bit, oppnår best resultater under optimale forhold. Dette skyldes at disse filterene henter ut mange flere tilgjenglige detaljer fra irisen under optimale forhold, noe som gjør at resultatene ved sammenligning blir bedre. Under støyfohold og uskarphetsforhold var det med 702-bit Haarfilter de beste resultatene ble oppnådd med. Under forandring av lysintensiteten i databasen var Log-Gabor filteret best ved 5 prosent økning og reduksjon, mens 702-bit Haarfilter var best ved en økning og reduksjon på 10 prosent. Ved rotasjon av bildene i databasen var det Log-Gabor filteret som ga best resultat. Dermed kan vi konkludere med at det er 702-bit Haarfilter vi totalt oppnådde best resultater med, og denne filteralgoritmen er best under ugunstige

forhold. Dette kan skyldes at dette filteret henter ut og koder færre men mer prominente, fremhevede og synlige detaljer fra irisen, og blir ikke så forstyrret av de elementene som bildene preges av under de simulerte forholdene. Dette gjør 702-bit Haarfilter til det beste filteret under de forskjellige forholdene som er simulert i denne oppgaven.

6 Forslag til videre arbeid

Underveis i arbeidet med masteroppgaven har det dukket opp flere idéer til videre arbeid med samme problemområdet det har blitt arbeidet med i denne oppgaven.

I dette prosjektet har det kun blitt brukt en segmenteringsalgoritmene (Hough transform) som har vist seg å være veldig bra under optimale forhold, men som er forholdsvis sensitiv under varierende ugunstige forhold. Siden denne delen av irisgjenkjenningen er så viktig vil det være naturlig å implementere andre segmenteringsalgoritmer. Et forslag vil være å implementere segmenteringsalgoritmen til Cho et al. [23] som er utviklet for å håndtere samme forhold som det har blitt arbeidet med i denne oppgaven. Disse algoritmene kan muligens både segmentere raskere og være med på å oppnå bedre resultater under slike forhold.

Når det gjelder filteralgoritmer kan det også være nyttig å implementere og teste 2D Gabor filteret som er utviklet av Daugman [7] og Zero-crossing filteret til Boles and Boashash [10], for å se hvordan disse er under ugunstige forhold. Det er også mulig å lage nye filteralgoritmer ved hjelp av Haar wavelet. Her vil det være naturlig å eksperimentere med forskjellige normaliserings-størrelser på irismønsteret, og kombinere forskjellige koeffisienter fra Haartransformen(vertikal, horisontal og diagonal) som også vil gi andre størrelser på irisvektorene. Et forslag vil være å benytte koeffisienter fra alle transformene som returneres(f.eks første, andre, tredje og fjerde transform). I 702-bit Haarfilteret dannes vektoren ved å kombinere vertikale, horisontale og diagonale koeffisienter fra fjerde og femte Haar transform. En ide kan være å ta med koeffisienene fra den tredje transformen også, slik at vektoren inneholder flere detaljer fra irisen. Videre kunne det vært interessant å ha kjørt de samme eksperimentene og sett effekten av å ta med flere koeffisienter fra Haar transformen. Et alternativ kan også være å implementere en metode som heter Wavelet Transform Modulus Maxima (WTMM) [27] som er i stand til å detektere og hente ut de mest atteraktive detaljene i et signal. Et forslag vil være å teste ut forskjellige normaliserings-størrelser på irismønsteret, for så å finne det lokale modulus maxima større enn en forhåndsatt terskelverdi for hvert subband som metoden transformerer.

Videre kan det være nyttig å teste algortimene i dette prosjektet mot en database bestående av ekte bilder som er tatt med mobiltelefon, PDA, webkamera osv for å belyse relevansen av de parameterene som er valgt i denne oppgaven. Det vil gjøre det mer realistisk i forhold til en ekte applikasjon der iris blir brukt til å verifisere ved innlogging. Her kan det også være nyttig å forhåndsprossesere bildene slik at irisdetaljene blir skarpere og kommer klarere frem i bildene. Dette kan f.eks gjøres ved å kjøre bildene gjennom en glattingsfunksjon (smoothing function) for så å se effekten av dette.

Bibliografi

[1] John Daugman. http://www.cl.cam.ac.uk/ jgd1000/.

[2] Oyvind W. Kvalsund. http://www.ii.uib.no, 2005.

[3] Libor Masek. Recognition of human iris patterns for biometric identification, 2003.

[4] Shinyoung Lim, Kwanyong Lee, Okhwan Byeon, and Taiyun Kim. Efficient iris recognition through improvement of feature vector and classifier. *ETRI Journal*, 23, No. 2:61–70, June 2001.

[5] Dr Brian Freeman. The how and why of the eye. *DIRECTION 2*, 2:10–16, School of Anatomy, University of NSW, Sydney, NSW 2052, Australia, 1999.

[6] Jaroslav Pospíil Ale Muroò. *THE HUMAN IRIS STRUCTURE AND ITS USAGES*. PhD thesis, Natural Science Faculty of Palacký University, Czech Republic, February 16, 2000.

[7] John Daugman. How iris recognition works. In *Proceedings of 2002 International Conference on Image Processing, Vol. 1*, pages 21–30, 2002.

[8] John Daugman. The importance of being random: statistical principles of iris recognition. In *Pattern Recognition Society.*, volume 36, pages 279–291, University of Cambridge, The Computer Laboratory, Cambridge, 2001.

[9] RICHARD P. WILDES. Iris recognition: An emerging biometric technology. In *Proceedings of the IEEE, Vol. 85, No. 9*, number 1348-1363, 1997.

[10] W. W. Boles and B. Boashash. A human identification technique using images of the iris and wavelet transform. In *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, volume 46, No. 4, pages 1185–1188, APRIL 1998.

[11] C. H. Daouk, L. A. El-Esber, F. D. Kammoun, and M. A. Al Alaoui. Iris recognition. In *IEEE ISSPIT*, pages 558–562, 2002.

[12] Sanchez-Avila, C. Sanchez, and Martin-RocheD. Iris recognition for biometric identification using dyadic wavelet transform zero-crossing. In *Proceedings of the IEEE 35th International. Camahan Conference on Security Technology*, pages 272 –277, 2001.

[13] Li Ma, Yunhong Wang, and Tieniu Tan. Iris recognition using circular symmetric filters. In *National Laboratory of Pattern Recognition*, pages 414–417, Institute of Automation, 2002.

[14] C. Tisse, L. Martin, L. Torres, and M. Robert. Person identification technique using human iris recognition. In *International Conference on Vision Interface*, pages 3–6, Canada, 2002.

[15] W. Kong and D. Zhang. Accurate iris segmentation based on novel reflection and eyelash detection model. In *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 263–266, Hong Kong, 2001.

[16] N. Ritter. Location of the pupil-iris border in slit-lamp images of the cornea. In *Proceedings of the International Conference on Image Analysis and Processing*, page 740, 1999.

[17] Bob Fisher et. al. http://www.cee.hw.ac.uk/hipr/html/hipr$_{t}$op.html, 1996.

[18] Tai Sing Lee. Image representation using 2d gabor wavelets. In *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, volume VOL. 18, pages 1–13, OCTOBER 1996.

[19] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. In *Journal of the Optical Society of America*, pages 2379–2394, 1987.

[20] Mayank Vatsa, Richa Singh, and P. Gupta. Comparison of iris recognition algorithms. In *IEEE, IClSlP*, pages 354–358, 2004.

[21] C. Sanchez-Avila, R. Sanchez-Reillo, and D. de Martin-Roche. Iris-based biometric recognition using dyadic wavelet transform. In *IEEE AESS Systems Magazine*, pages 3–6, October 2002.

[22] OKI. Oki introduces japan's first iris recognition for camera-equipped mobile phones and pdas. http://www.oki.com/en/press/2006/z06114e.html, Oki Electric Industry Co. 2006.

[23] Dal Ho Cho, Kang Ryoung Park, and Dae Woong Rhee. Real-time iris localization for iris recognition in cellular phone. In *Proceedings of the Sixth International Conference on Software Engineering*, 2005.

[24] CASIA iris image database. http://www.cbsr.ia.ac.cn/IrisDatabase.htm.

[25] LEI iris database. http://www.lei.lt/main.php?m=275&k=9.

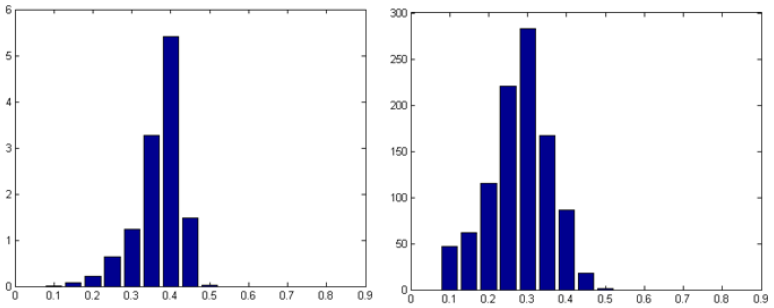
[26] UBIRIS Database. http://iris.di.ubi.pt/.

[27] C. Li, C. Zheng, and C. Tai. Detection of ecg characteristic points using wavelet transform. In *IEEE Transactions on Biomedical Engineering*, volume 52, 1995.

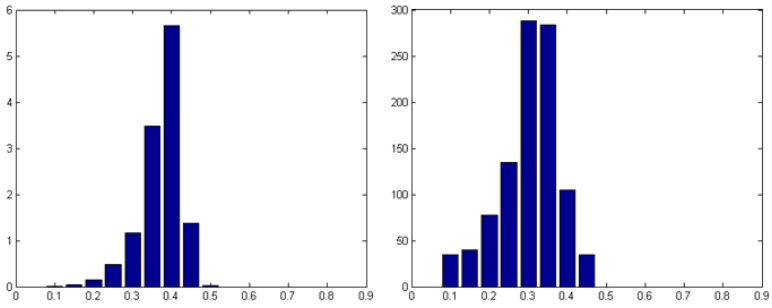
A Hammingfordeling under støyforhold

Til venstre vises interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

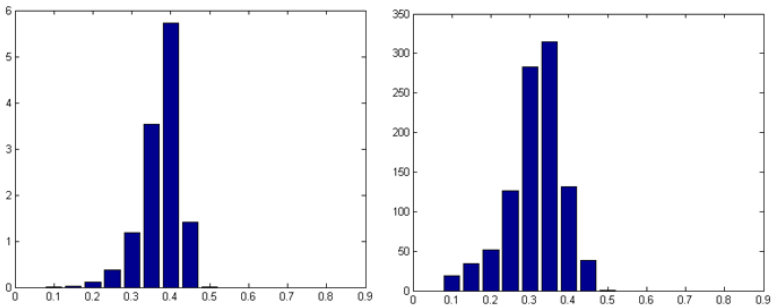
A.1 Hammingfordeling med 702-bit Haarfilter



Figur 46: Hammingfordelingen med 702-bit Haarfilter, med støyvarianse 0.002

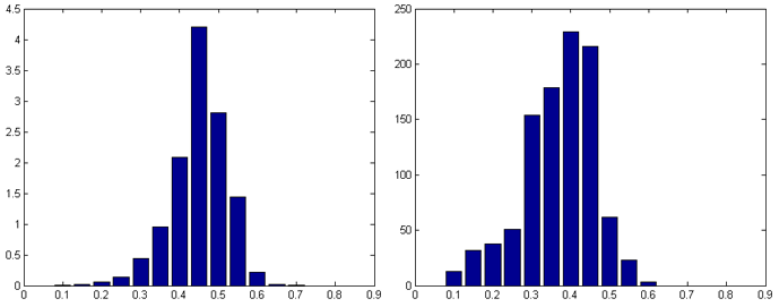


Figur 47: Hammingfordelingen med 702-bit Haarfilter, med støyvarianse 0.004

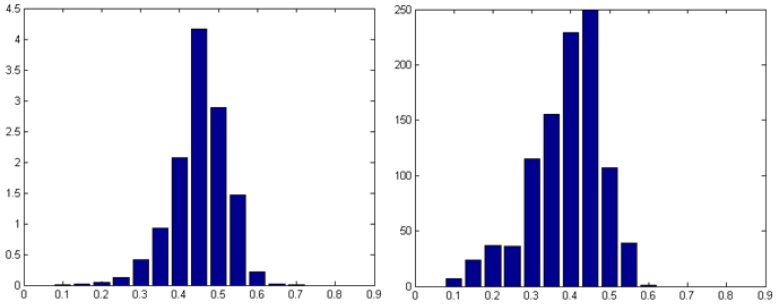


Figur 48: Hammingfordelingen med 702-bit Haarfilter, med støyvarianse 0.006

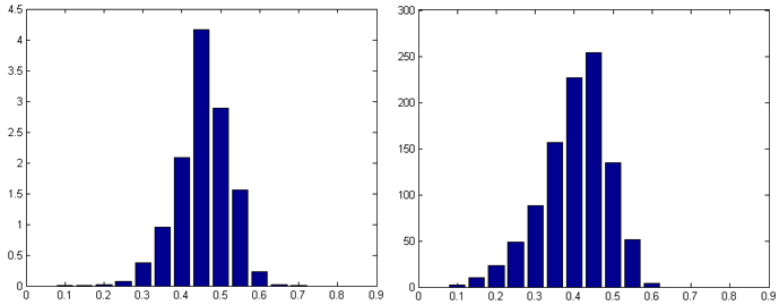
A.2 Hammingfordeling med 87-bit Haarfilter



Figur 49: Hammingfordelingen med 87-bit Haarfilter, med støyvarianse 0.002

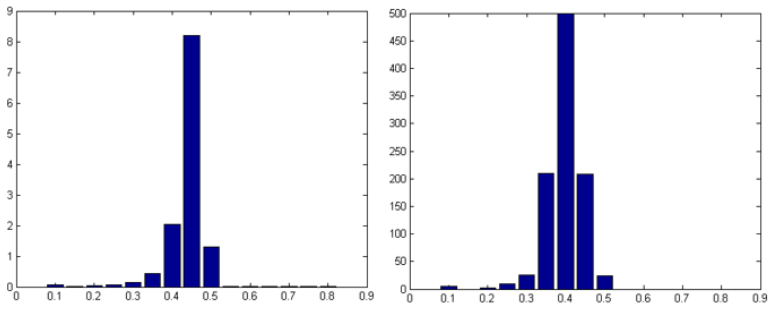


Figur 50: Hammingfordelingen med 87-bit Haarfilter, med støyvarianse 0.004

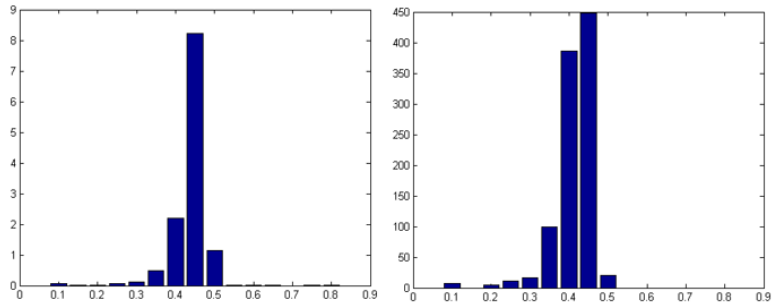


Figur 51: Hammingfordelingen med 87-bit Haarfilter, med støyvarianse 0.006

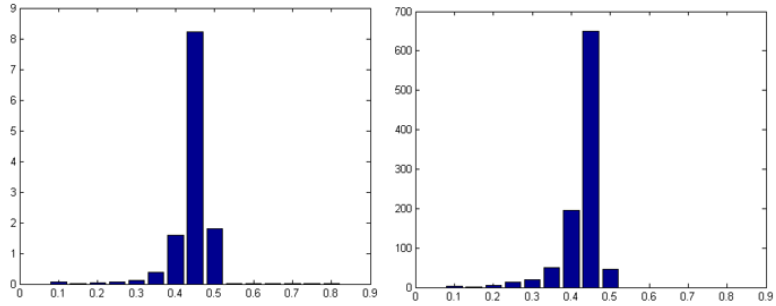
A.3 Hammingfordeling med Log-Gabor filter



Figur 52: Hammingfordelingen med Log-Gabor, med støyvarianse 0.002

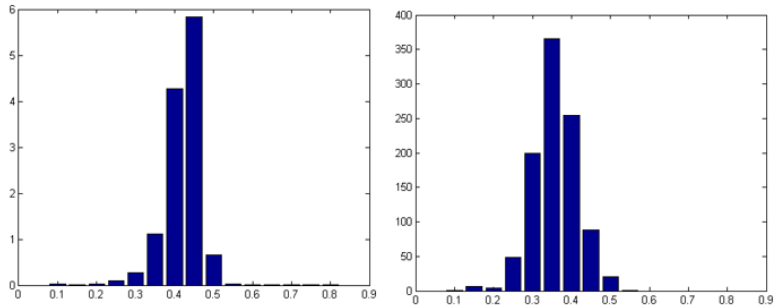


Figur 53: Hammingfordelingen med Log-Gabor, med støyvarianse 0.004

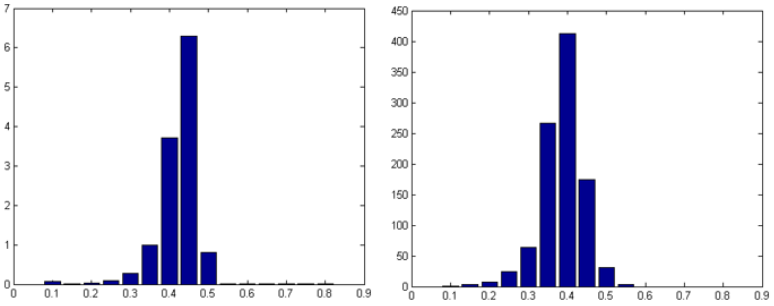


Figur 54: Hammingfordelingen med Log-Gabor, med støyvarianse 0.006

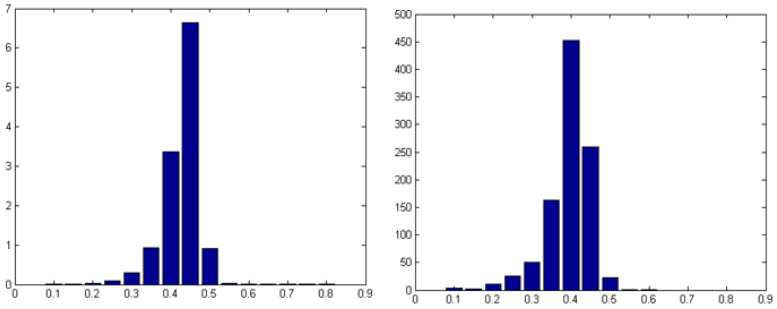
A.4 Hammingfordeling med Laplacian of Gaussian filter



Figur 55: Hammingfordelingen med Laplacian of Gaussian, med støyvarianse 0.002



Figur 56: Hammingfordelingen med Laplacian of Gaussian, med støyvarianse 0.004

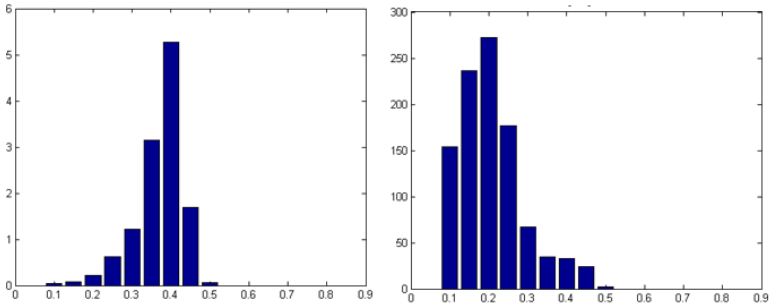


Figur 57: Hammingfordelingen med Laplacian of Gaussian, med støyvarianse 0.006

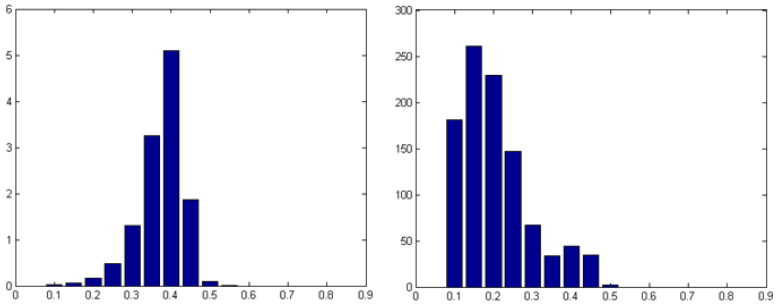
B Hammingfordeling under blurforhold

Til venstre vises interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

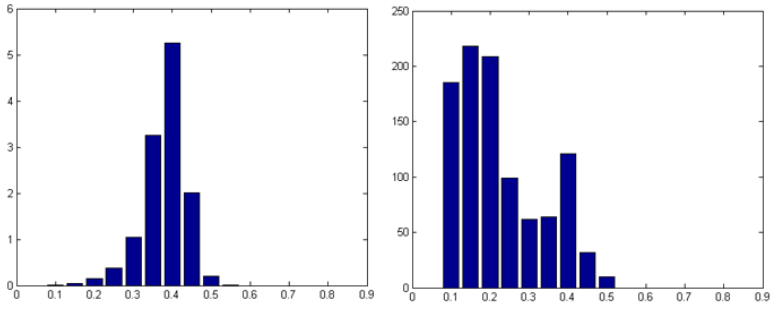
B.1 Hammingfordeling med 702-bit Haarfilter



Figur 58: Hammingfordelingen med 702-bit Haarfilter, med blur radius på 2

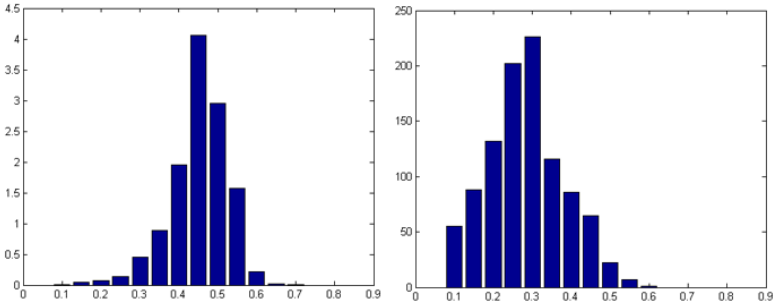


Figur 59: Hammingfordelingen med 702-bit Haarfilter, med blur radius på 4

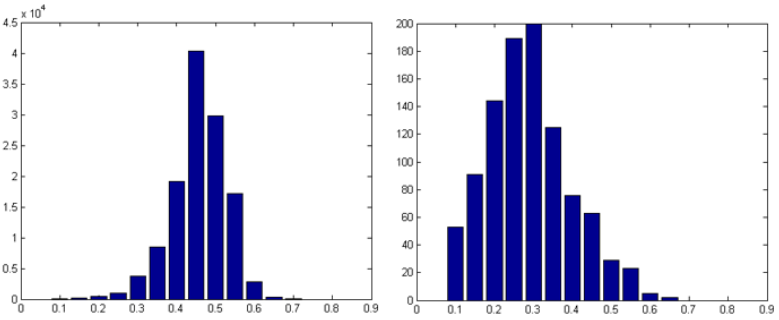


Figur 60: Hammingfordelingen med 702-bit Haarfilter, med blur radius på 6

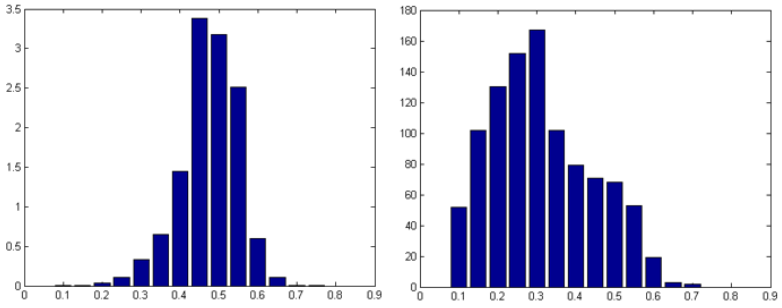
B.2 Hammingfordeling med 87-bit Haarfilter



Figur 61: Hammingfordelingen med 87-bit Haarfilter, med blur radius på 2

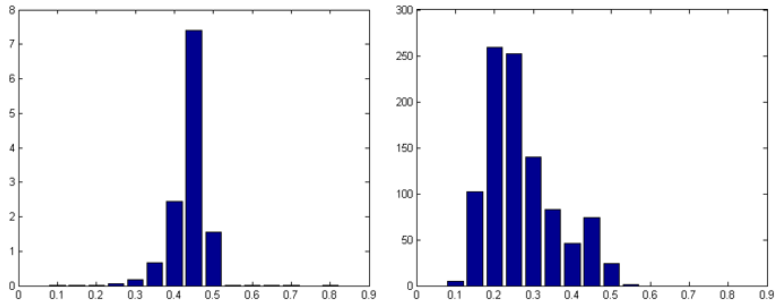


Figur 62: Hammingfordelingen med 87-bit Haarfilter, med blur radius på 4

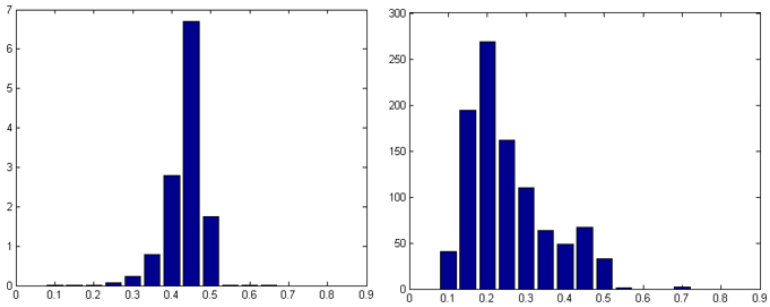


Figur 63: Hammingfordelingen med 87-bit Haarfilter, med blur radius på 6

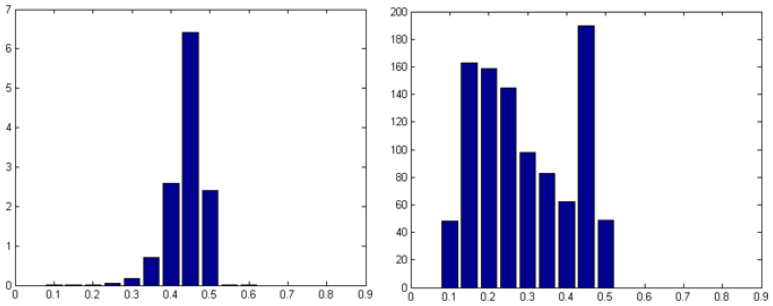
B.3 Hammingfordeling med Log-Gabor filter



Figur 64: Hammingfordelingen med Log-Gabor, med blur radius på 2

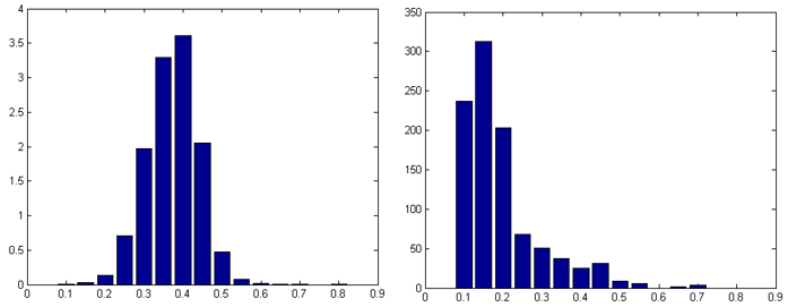


Figur 65: Hammingfordelingen med Log-Gabor, med blur radius på 4

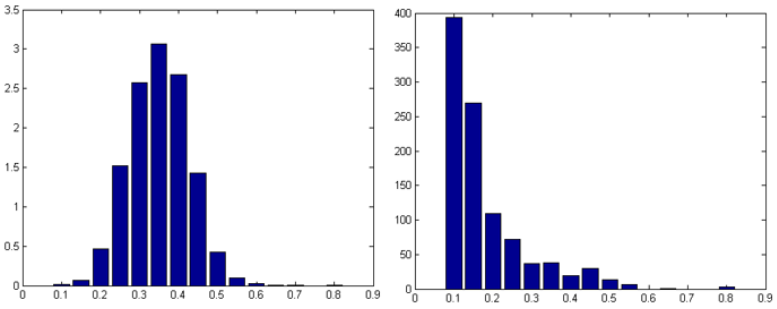


Figur 66: Hammingfordelingen med Log-Gabor, med blur radius på 6

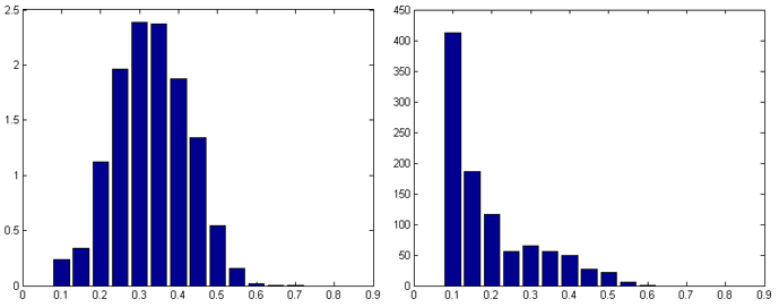
B.4 Hammingfordeling med Laplacian of Gaussian filter



Figur 67: Hammingfordelingen med Laplacian of Gaussian, med blur radius på 2



Figur 68: Hammingfordelingen med Laplacian of Gaussian, med blur radius på 4

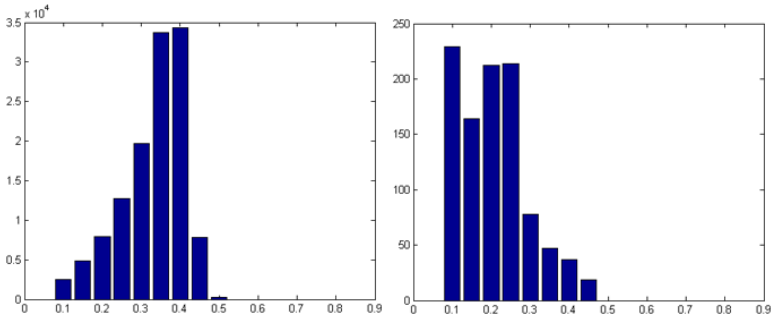


Figur 69: Hammingfordelingen med Laplacian of Gaussian, med blur radius på 6

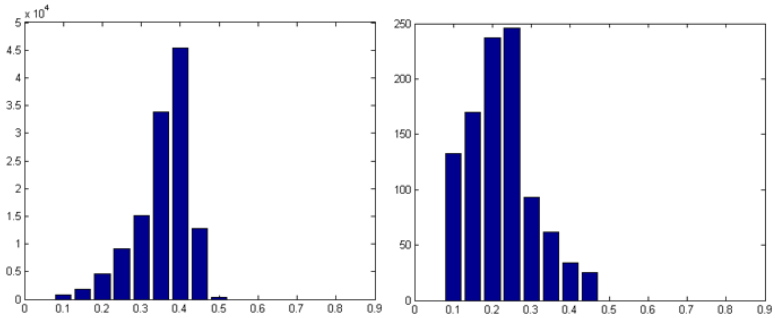
C Hammingfordeling under lysforandring

Til venstre vises interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

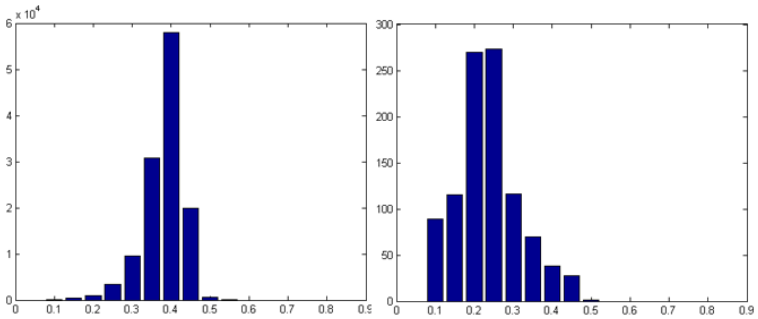
C.1 Hammingfordeling med 702-bit Haarfilter



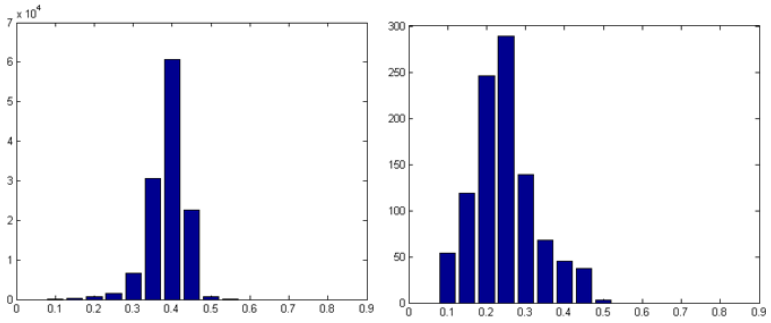
Figur 70: Hammingfordelingen med 702-bit Haarfilter, med -10 prosent



Figur 71: Hammingfordelingen med 702-bit Haarfilter, med -5 prosent

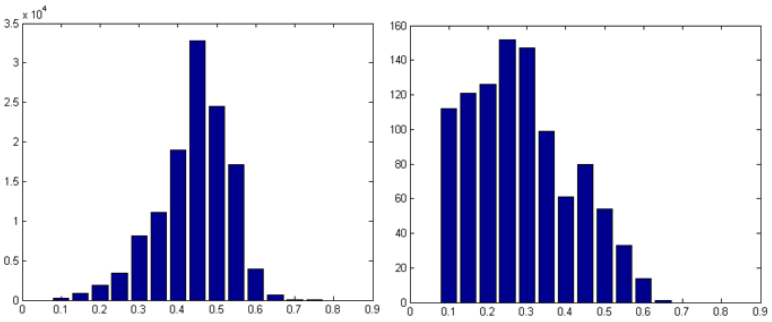


Figur 72: Hammingfordelingen med 702-bit Haarfilter, med +5 prosent

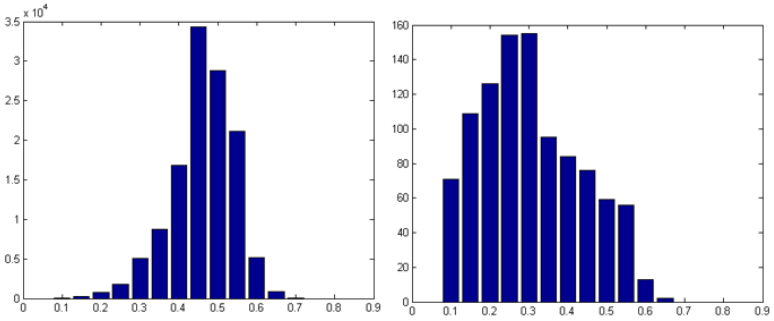


Figur 73: Hammingfordelingen med 702-bit Haarfilter, med +10 prosent

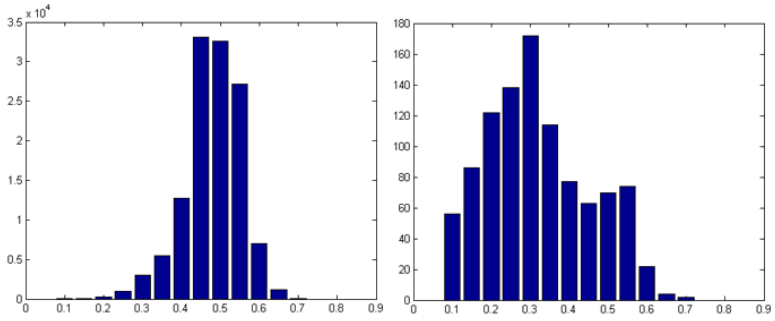
C.2 Hammingfordeling med 87-bit Haarfilter



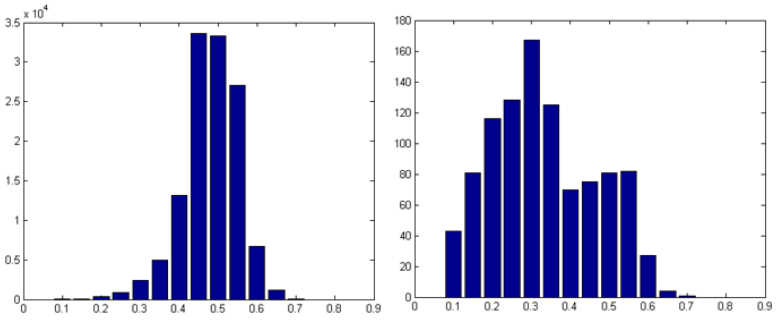
Figur 74: Hammingfordelingen med 87-bit Haarfilter, med -10 prosent



Figur 75: Hammingfordelingen med 87-bit Haarfilter, med -5 prosent

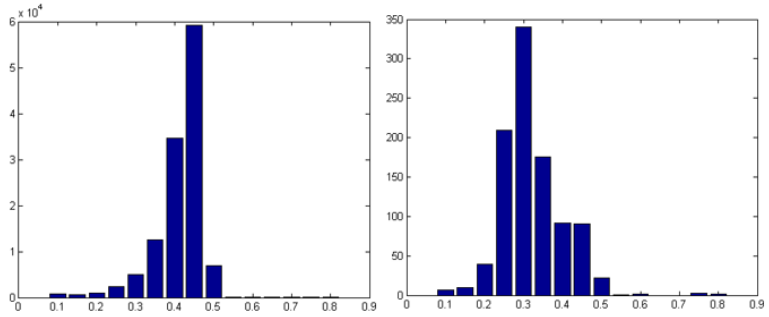


Figur 76: Hammingfordelingen med 87-bit Haarfilter, med +5 prosent

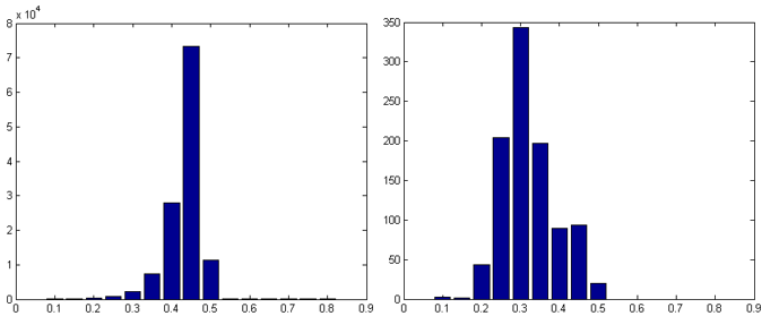


Figur 77: Hammingfordelingen med 87-bit Haarfilter, med +10 prosent

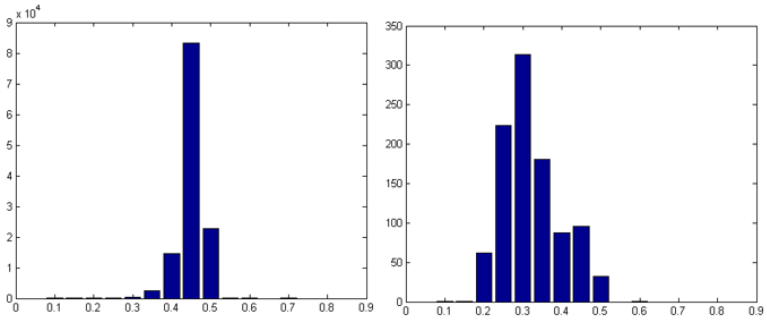
C.3 Hammingfordeling med Log-Gabor filter



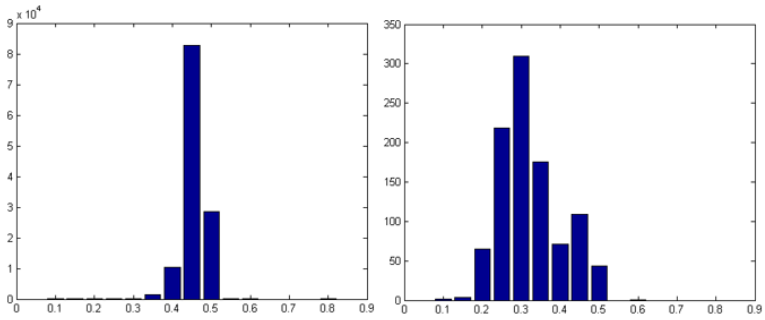
Figur 78: Hammingfordelingen med Log-Gabor, med -10 prosent



Figur 79: Hammingfordelingen med Log-Gabor, med -5 prosent

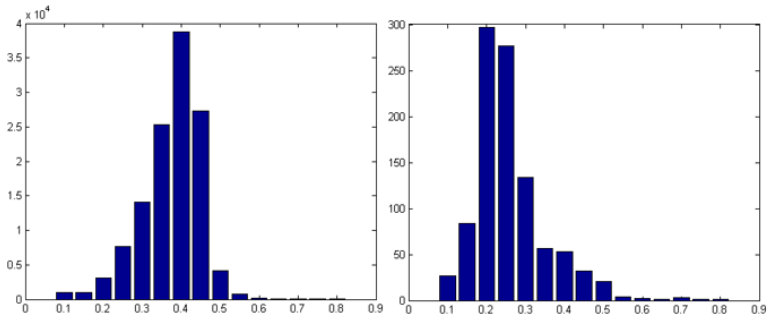


Figur 80: Hammingfordelingen med Log-Gabor, med +5 prosent

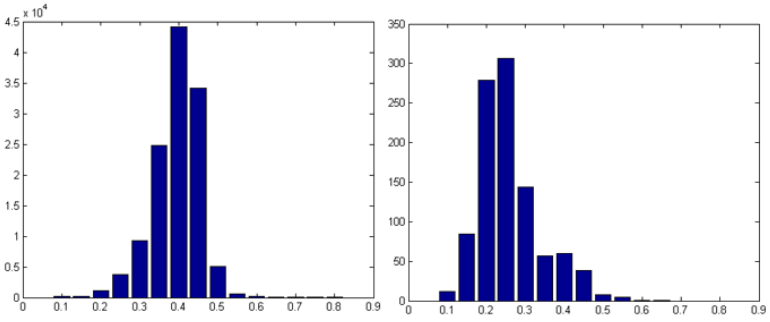


Figur 81: Hammingfordelingen med Log-Gabor, med +10 prosent

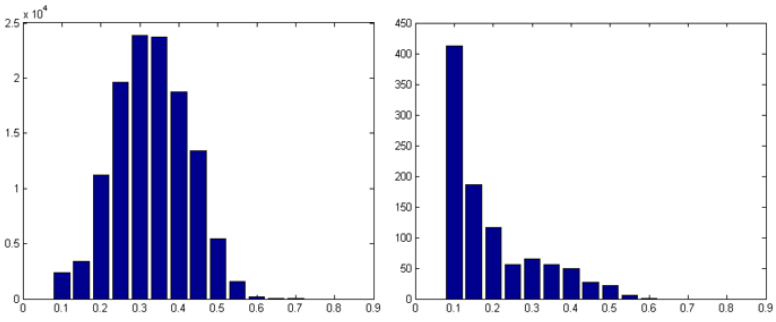
C.4 Hammingfordeling med Laplacian of Gaussian filter



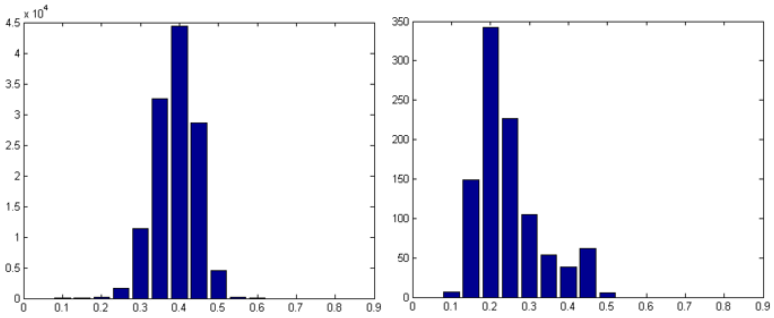
Figur 82: Hammingfordelingen med Laplacian of Gaussian, med -10 prosent



Figur 83: Hammingfordelingen med Laplacian of Gaussian, med -5 prosent



Figur 84: Hammingfordelingen med Laplacian of Gaussian, med +5 prosent

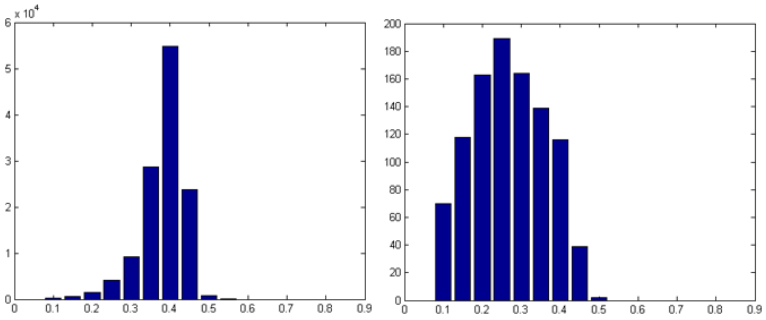


Figur 85: Hammingfordelingen med Laplacian of Gaussian, med +10 prosent

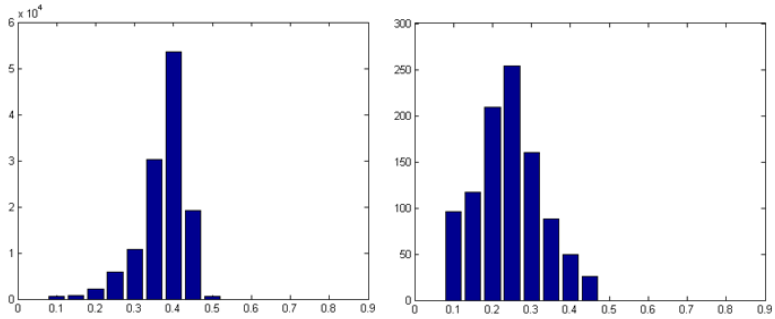
D Hammingfordeling under rotasjonsforhold

Til venstre vises interklasse-sammenligningen og til høyre intraklasse-sammenligningen.

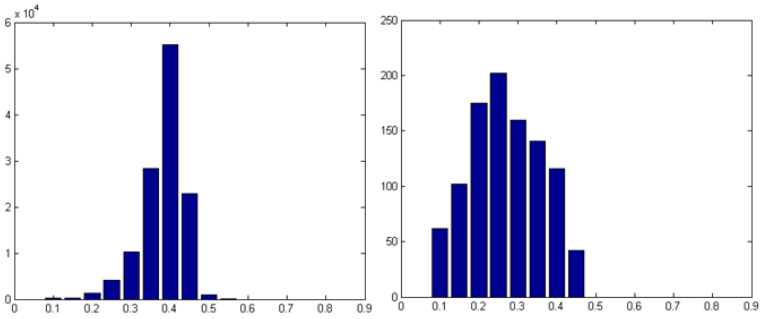
D.1 Hammingfordeling med 702-bit Haarfilter



Figur 86: Hammingfordelingen med 702-bit Haarfilter, med 2 grader rotasjon

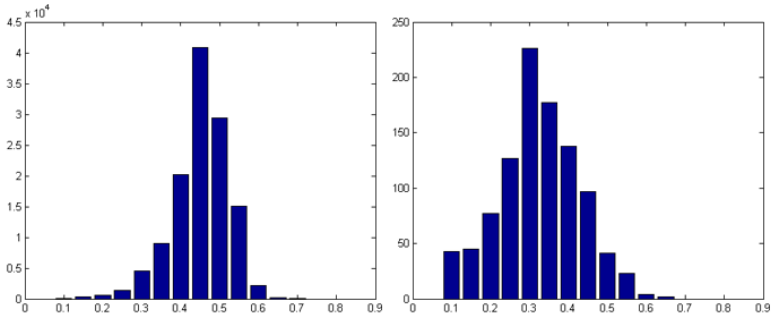


Figur 87: Hammingfordelingen med 702-bit Haarfilter, med 3 grader rotasjon

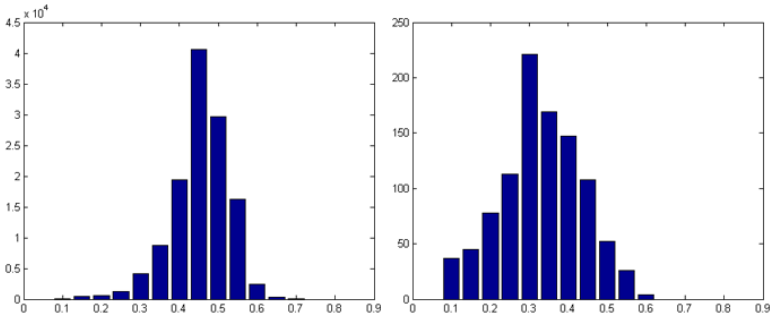


Figur 88: Hammingfordelingen med 702-bit Haarfilter, med 4 grader rotasjon

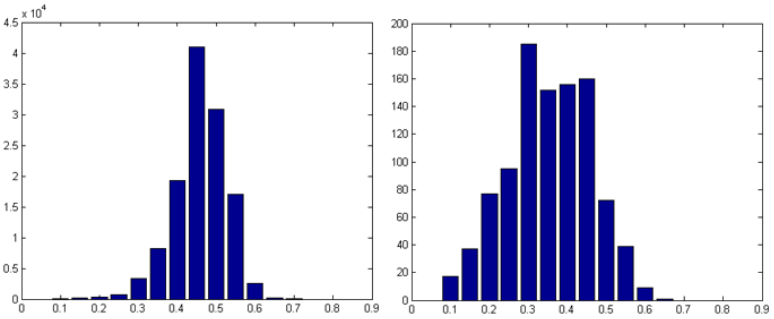
D.2 Hammingfordeling med 87-bit Haarfilter



Figur 89: Hammingfordelingen med 87-bit Haarfilter, med 2 grader rotasjon

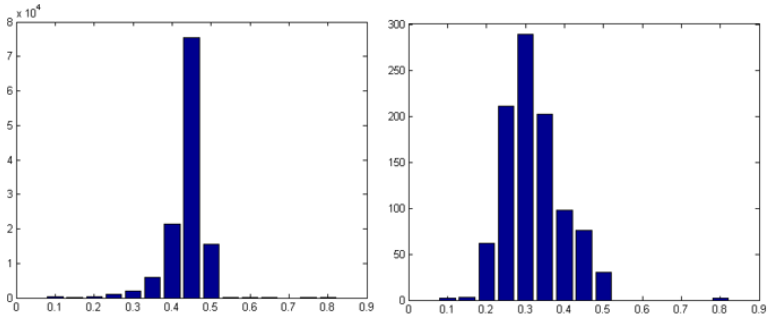


Figur 90: Hammingfordelingen med 87-bit Haarfilter, med 3 grader rotasjon

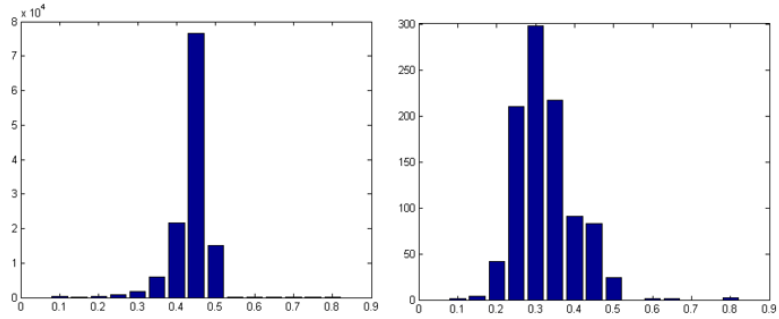


Figur 91: Hammingfordelingen med 87-bit Haarfilter, med 4 grader rotasjon

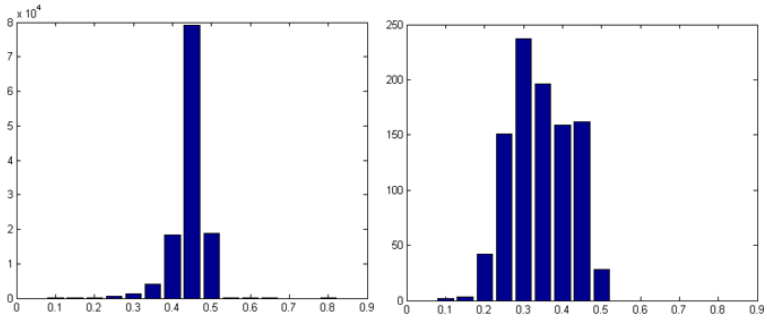
D.3 Hammingfordeling med Log-Gabor filter



Figur 92: Hammingfordelingen med Log-Gabor, med 2 grader rotasjon

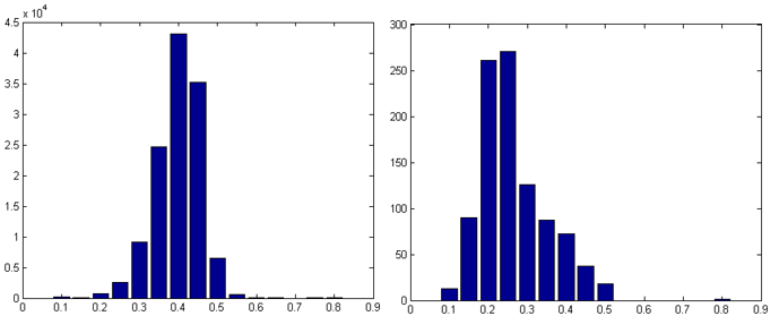


Figur 93: Hammingfordelingen med Log-Gabor, med 3 grader rotasjon

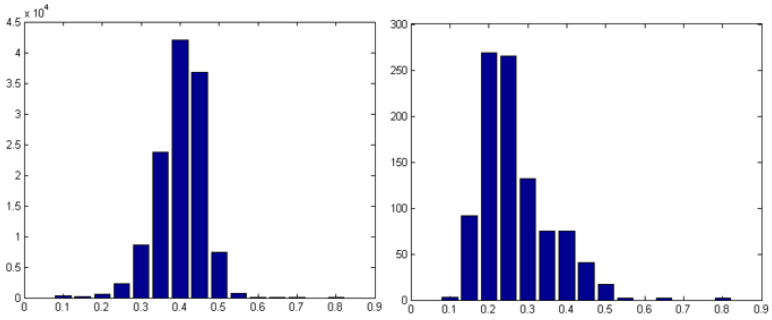


Figur 94: Hammingfordelingen med Log-Gabor, med 4 grader rotasjon

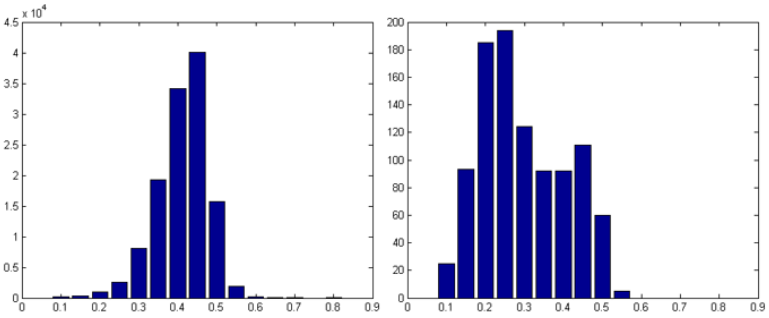
D.4 Hammingfordeling med Laplacian of Gaussian filter



Figur 95: Hammingfordelingen med Laplacian of Gaussian, med 2 grader rotasjon



Figur 96: Hammingfordelingen med Laplacian of Gaussian, med 3 grader rotasjon



Figur 97: Hammingfordelingen med Laplacian of Gaussian, med 4 grader rotasjon