Database for bildediagnostisk programvare for hudkreft

Mastergradsoppgave

av

Øyvind Westgaard Kvalsund

Veileder

Dhayalan Velauthapillai

Institutt for Informatikk

Universitetet i Bergen

Vår 2008

Forord

Denne rapporten er laget som det avsluttende arbeidet på mastergraden i informatikk ved Universitetet i Bergen.

Jeg vil benytte anledningen til først og fremst til å takke min veileder Dhayalan Velauthapillai for god støtte underveis i arbeidet og verdifulle kommentarer på rapporten.

Jeg vil også rette en takk til Kristian P. Nilsen, Børge Hamre, Lu Zhao, Simen Malmin og de andre i Balter Medical som har vært til god hjelp for å sette seg inn i den terminologien som har nødvendig for å kunne utføre oppgaven.

Jeg vil takke Jan Martin Langeland og Svein Even Vikshåland for gode faglige diskusjoner underveis i oppgaven. Jeg vil også takke Carsten Helgesen og Martin Lie for gjennomgang av databaseløsningen.

Sist men ikke minst vil jeg takke familien min for deres støtte.

Øyvind Westgaard Kvalsund

Bergen, Våren 2008

Innhold

1 Introduksjon

Å ta bilder av kroppen for å kunne se under overflaten på huden er en gammel idé. At ulike typer kroppsvev reagerer ulikt når en sender lys eller stråling utnyttes i mange medisinske redskaper i dag. Røntgenbilder, magnetresonanstomografi (MRI), ultralyd og positronemisjonstomografi (PET) er anerkjente metoder som gir resultater i bilder som kan vurderes av medisinsk personell. Utstyret som brukes er i mange tilfeller dyrt og er bare tilgjengelig på sykehus, ikke på de lokale legekontorene.

1.1 Historisk bakgrunn for hudkreft

Kroppen er bygget opp av celler som har ulike egenskaper avhengig av hvilken funksjon de skal utføre. Nye celler produseres hele tiden for å erstatte celler som er blitt skadet eller som er døde. De nye cellene lages ved celledeling. Vanlig celledeling skjer ved at en celle dobler arvestoffet DNA og deler seg i to. Diagnosen kreft stilles når man får ukontrollert celledeling, der cellene ikke utfører de oppgavene den normalt skulle gjøre. Når kreftceller hoper seg opp på samme sted dannes en kreftsvulst. Det er fare for spredning når celler bryter løs fra kreftsvulster og føres med lymfeknuter eller blodårer og sprer seg til vev andre steder i kroppen [1].

Malignt melanom er en form for hudkreft der kreftsvulsten oppstår i hudens pigmentceller. Ikke alle detaljer omkring årsakene til føflekkreft er forstått, men man vet det har sammenheng med at huden overeksponeres for UV (ultrafiolett) lys [2]. Malignt melanom rammer over 1000 mennesker i Norge hvert år. Antall diagnostiserte tilfeller har seksdoblet seg de siste 30 år [3], men de siste årene har antall tilfeller på årsbasis begynt å flate ut. Blant årsakene til økningen er at det siden 1950-tallet er blitt mer moteriktig med solbrun hud. Helge- og feriesoling der en blir utsatt for store doser med sollys på kort tid knyttes til økende forekomst av hudkreft [1]. UV-stråling absorberes av DNA-molekylene i hudcellene og kan skade dem direkte [2].

Å få diagnostisert sykdommen tidlig er en nøkkelfaktor for vellykket behandling. Dess lenger tid det går før sykdommen blir behandlet jo større sjanse er det for spredning [4].

Forskere er stadig ute etter nye metoder for å diagnostisere hudkreft på en effektiv, rask og enkel måte. Balter Medical er et firma som startet i Bergen, men er registrert i USA. Forsking og utvikling foregår i avdelingen av Balter Medical som er lokalisert i Bergen. Balter Medical har utviklet et nytt diagnostiseringsverktøy for å påvise den mest dødelige formen for hudkreft, malignt melanom. De kan så langt kan vise til meget lovende resultater. Forslag til diagnose gjøres ved bildediagnostisk analyse. For å kunne gjøre denne analysen trenges det store mengder bilder.

Målet med dette mastergradsprosjektet er å utvikle en database som lagrer bilder, data knyttet til pasienter og relasjonene mellom bilder og pasienter.

1.2 Oppbygging av rapporten

Rapporten begynner med å sammenligne nåværende metoden med den nye metoden for å kunne diagnostisere føflekker i kapittel 2. Kapittel 3 tar for seg oppbyggingen av diagnostiseringsverktøyet for den nye metoden. En vil se hvilket verktøy legen benytter seg av og få et innblikk i prosessen for å fremstille en diagnose. Kapittel 4 forteller litt mer om diagnostiseringsprosessen for den nye metoden. Det blir fortalt om dagens løsning internt i bedriften for lagring og spørring på data, og problemene med dem. Kapittel 5 analyserer problemene som skal løses i denne oppgave, hvilken metode de skal løses på, og hvilke krav som settes til løsningen og systemet. Kapittel 6 går gjennom de delene av den nåværende løsningen som skal utbedres. Den vil videre gå inn på hvordan de ulike delene av problemet er blitt løst. Kapittel 7 viser implementasjonen av programmet. Her får man eksempler på brukerens opplevelse av programmet. Neste kapittel har en evaluering av hvordan systemet er bygget opp og begrunner en del av valgene som ble tatt underveis. Avslutningen av rapporten er en oppsummering og forslag til videre arbeid.

Appendiks Begrepsliste som finnes i slutten av oppgaven er ment for å kunne hjelpe om det er enkelte ord som er uklare.

2 Sammenligning av diagnostiseringsmetoder

For å få vurdert en føflekk gjøres det en undersøkelse av føflekken hos en allmennpraktiserende lege eller en spesialist. Om legen vurderer føflekken som mistenksom, vil man foreta en biopsi det vil si skjære føflekken bort og sende den til en patolog. Patologen undersøker biopsiprøven ved å tilsette farge, dele den opp i tynne flak og studere den under mikroskop. Målet med den nye diagnostiseringsmetoden er å gi den allmennpraktiserende lege eller spesialist et bedre verktøy til å vurdere føflekken uten å ta biopsi.

Statistisk viser det seg at bare 1 av 40 tilfeller som blir undersøkt av patologer er ondartet [5]. Kostnaden med å undersøke en føflekk ved hjelp av den nye diagnostiseringsmetoden er en brøkdel (1/50) av kostnaden med å foreta en biopsi [5].

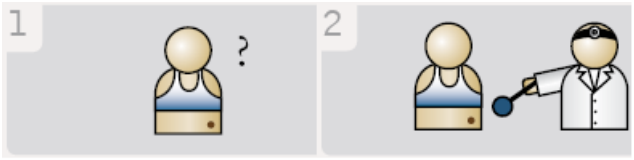
2.1 Eksisterende diagnostiseringsmetode

Figur 2.1 Diagnostisering med nåværende metode [5]

Figur 2.1 viser diagnostiseringsprosessen for en pasient i dag.

1. Pasienten oppdager en føflekk som ser mistenkelig ut.
2. Legen studerer føflekken visuelt.
3. Om legen er i tvil vil han i de fleste tilfeller skjære bort føflekken, noe som vil kunne etterlate et arr på pasienten.
4. Føflekken blir skåret ut og sendt til en patolog for nærmere undersøkelse.
5. Patologen leverer diagnose til pasienten.

2.2 Den nye diagnostiseringsmetode



Figur 2.2 Diagnostisering med ny metode [5]

1. Pasienten oppdager føflekk som ser mistenkelig ut.
2. Legen skanner føflekken og kan etter bildeprosessering gi en diagnose på stedet.

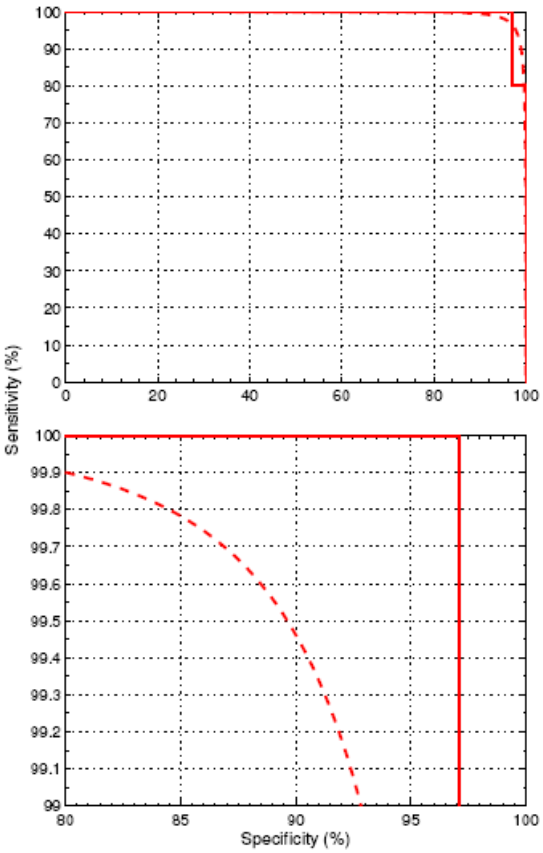
Diagnosen gjøres ved å analysere bildene med en patentert metode utviklet av Balter Medical. Ved å prosessere bildene kan legen dessuten få presentert en rekke parametre som hjelper til med diagnostiseringen:

- 3D morfologisk (form) informasjon
- Oksygenprosent i blodet
- Konsentrasjon av blod i lærhuden (dermis)
- Konsentrasjon av keratin i hornlaget (stratum corneum)
- Fordelingen av melanosomer i overhuden (epidermis)
- Størrelsesfordelingen til melanosomer i overhuden (epidermis)

Disse parameterne blir brukt til å kalkulere en diagnoseindeks som er en indikator på hvorvidt melanomet er godartet eller ondartet.

2.3 Sammenligning av diagnostiseringsmetoder

Som nevnt i innledningen viser bare 1 av 40 biopsiprøver seg å være ondartet melanom. Disse tallene viser at visuell inspeksjon av en føflekk ikke er nøyaktig nok for å stille en god diagnose. Dette kan skyldes at legene vil være på den sikre siden, og la sjansen for at ikke fjerner en ondartet føflekk være liten. Pasientene kan vegre seg for å få sjekket føflekker av frykt for arrdannelse der føflekken har vært. Ventetid for diagnose er en ubehagelig tid for pasienten.



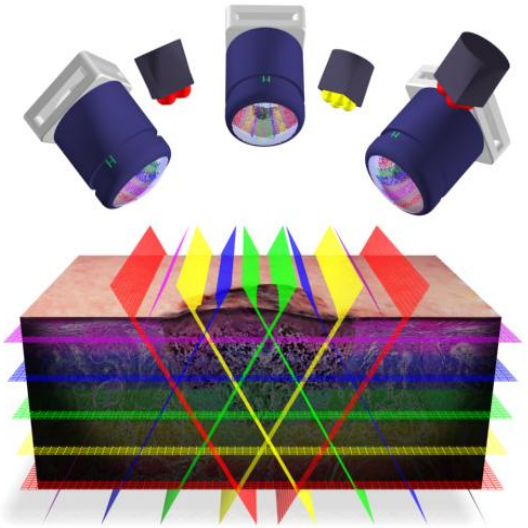
Figur 2.3 ROC kurve [5]. Stiplet linje viser forventete verdier, heltrukket linje viser reelle verdier på dagens datasett. Nederste ROC kurve viser et utdrag av den over.

Figur 2.3 viser en ROC (Receiver Operating Characteristic) kurve plottet med spesifisitet mot sensitivitet for melanomene Balter Medical har undersøkt til nå. Sensitivitet er sannsynlighet for positivt testresultat gitt positiv diagnose. Spesifisitet er sannsynlighet for negativt testresultat gitt negativ diagnose. Balter Medical ønsker ca 99 % sensitivitet og ca 90 % spesifisitet. Sensitiviteten på dataene de har analysert til nå er 100 %, spesifisiteten er 94 %. Omfattende klinisk utprøving trengs før den endelige kvaliteten på metodene er kjent.

Å sammenligne sensitivitet og spesifisitet som allmennpraktiserende leger og spesialister leverer i dag mot den nye løsningen, blir vanskelig. Leger vil i dag i de fleste tilfeller skjære bort føflekker dersom det i det hele tatt skulle kunne være tvil om at det kan være et ondartet melanom. Det blir ikke gjort biopsi på føflekker som ikke blir regnet som skumle, så å beregne spesifisitet er vanskelig. Om man skal beregne spesifisitet ut fra at 1 av 40 biopsi prøver som blir undersøkt er ondartede melanomer vil det gi en spesifisitet på 2,5 %.

3 Oppbyggingen av diagnostiseringsverktøy

Diagnostisering ved hjelp av avbilding er basert på at normal hud, godartede – og ondartede lesjoner har forskjellige optiske egenskaper, og dermed vil ha ulike reflektanser når man belyser dem med lys med ulike bølgelengder. Bølgelengdene på lyset ligger fra UV (ultrafiolett) lys (350nm) som belyser bare det overfladiske i huden til NIR (nær infrarødt) lys (1000 nm) som penetrerer flere millimeter inn i huden [5].



Figur 3.1 Lamper og kamera i skanner [5]

Når man skal undersøke en lesjon bruker legen en skanner. Skanneren inneholder kameraer og lyskilder som sitter i faste posisjoner i et lukket system. Kameraene og lysene er rettet mot en gjennomsiktig glassplate som legges mot lesjonen man ønsker å undersøke.

Når man skal undersøke en lesjon blir den belyst med 10 ulike LED (Light Emiting Diode) lyskilder som sender ut lys med ulike bølgelengder. Det blir tatt bilde av det lyset som reflekteres med tre CCD (Charged-Coupled Device) eller tre CMOS (Complementary Metal–Oxide–Semiconductor) kameraer som er plassert med tre ulike innfallsvinkler. Systemet er montert som vist på Figur 3.1.

Når man tar bilder av en lesjon hos en pasient vil dette generere 30 bilder, ett bilde med hvert kamera for hver bølgelengde, som blir lagret for videre analysering.

3.1 Kalibrering av pasientbilder

For å regne råbildene fra skanneren om til reflektans bilder må man kalibrere bildene. Dette gjøres ved å ta bilder av en kalibreringsstandard (CS) i tillegg til bildene av hudlesjonene. Dessuten tas ett sett med mørkestrømsbilder (DC). For å gjøre pasientbildene mer nøyaktig har Balter Medical utviklet en metode for å få luket vekk uønsket støy på pasientbildene.

Legen tar minst en gang daglig to sett med kalibreringsbilder, CS (Calibration Standard) og DC (Dark Current) bilder. For hvert av CS og DC bildesettene tar skanneren 10 bilder med hvert kamera for hver bølgelende, noe som gir et resultat på 300 bilder for hvert av bildesettene.

CS bildene blir tatt mot en helt hvit brikke som har egenskapen at den gir opp til 99 % reflektans (reflektert stråling) på lys som ligger mellom UV- til NIR bølgelengder. Denne egenskapen brukes til å kalibrere pasientbildene.

DC bilder er mørkestrømsbilder. Mørkestrømsbildene blir tatt ved å dekke glassplaten på skanneren slik at alt lys fra det lukkede systemet med lamper og kamera stenges ute. Bildene man får da viser bakgrunnsstøy fra kameraet.

Når man skal kalibrere data fra skannere der CMOS kameraer brukers, trengs det enda ett sett med DC bilder. Disse DC bildene blir referert til som DC2 (Dark Current 2) bilder. DC2 blir sammen med DC brukt for å identifisere fast støy hos skanneren.

Bedre spesifisert er målet med kalibreringsrutinen å produsere bilder av hudens reflektans, som er uavhengig av den opprinnelige lysfordelingen i bildet og eksponeringstidene til kameraene. De ukalibrerte pasientbildene som blir tatt av skanneren, blir referert til som ”level 0 – bilder”. De kalibrerte pasientbildene som blir laget i kalibreringsprosessen blir kalt ”level 1 – bilder”.

Balter Medical har utviklet analysemetoder for å se på formen til føflekker (morfologi analyse), samt se på fysiologiske faktorer. Algoritmene regnes som en foretningshemmelighet så oppgaven kommer ikke til å utdype hvordan man får fremstilt fysiologi- og morfologi dataene.

Nøyaktig hvilke bølgelengder og vinkler som benyttes for kamera og lamper, samt data knyttet til hvilke leger som har utført forsøkene, regnes som forretningshemmeligheter og har derfor blitt endret i denne oppgaven.

3.2 Resultat fra diagnostiseringsprosessen

3.2.1 Fysiologi

Om man vil sette seg inn i fysikken som ligger bak dette, kan det være til hjelp å lese nærmere på [6] og [7]

- Oksygenprosent i blodet.
  Blodet kan ha to tilstander. Det kan være oksygenert, blodet er da oksygenrikt, eller deoksigenert som er oksygenfattig blod. En lesjon i huden kan føre til skader på kapillærene (de fineste karforgreninger) noe som gjør at blodet i det området blir deoksygenert. Forskjell på nivået på oksygenprosent innen en lesjon kan være et tegn på at det er hudkreft.
- Konsentrasjon av blod i lærhuden (dermis).
  Det blir opprettet nye blodårer i området rundt skadet vev for å transportere blodet, og man vil da i noen tilfeller kunne se forskjell på blodkonsentrasjonen i friskt vev og i vev som inneholder melanom. Uregelmessig distribuering av blod i huden kan være et tegn på hudkreft.
- Konsentrasjon av keratin i hornlaget (stratum corneum)
  Lesjoner med melanom vil typisk ha lavt innhold av keratin. Om lesjonen er av type seborroisk keratose som er en godartet svulst vil det være høy forekomst av keratin. Felles for både melanom og seborroisk keratose er at de begge kan oppfylle de såkalte ABCD kriteriene som vil bli forklart i kapittel 3.2.2, men er det lav konsentrasjon av keratin er det et tegn på at lesjonen er et melanom.
- Fordelingen av melanosomer i overhuden (epidermis)
  Melanosomer finnes vanligvis i melanincellene i overhuden. Om man finner melanosomer, som ligger i andre deler av huden enn de vanligvis finnes, kan det være et tegn på at det er kreftceller som har spredd seg til området rundt, især om fordelingen av disse er uregelmessig.

3.2.2 Morfologi

Balter Medical har utviklet algoritmer som analyserer morfologiegenskapene i melanomene med hensyn på ABCD kriteriene [8]:

- **Asymmetri**. Melanomer er ofte asymmetriske. På godartede føflekker vil ene halvdelen av føflekken være samme størrelse og ha lik symmetri.
- **Border** (grense). Melanomer har ofte uregelmessig og uskarp avgrensing.
- **Color** (farge). Godartede føflekker kan ha forskjellige farger, men vil ofte bare ha en farge. Malignt melanom har nesten alltid flere fargenyanser.
- **Diameter**. Melanomer fortsetter å vokse. Føflekker med kreft er ofte store, med diameter over 6 millimeter.

3.2.3 Oppsummering

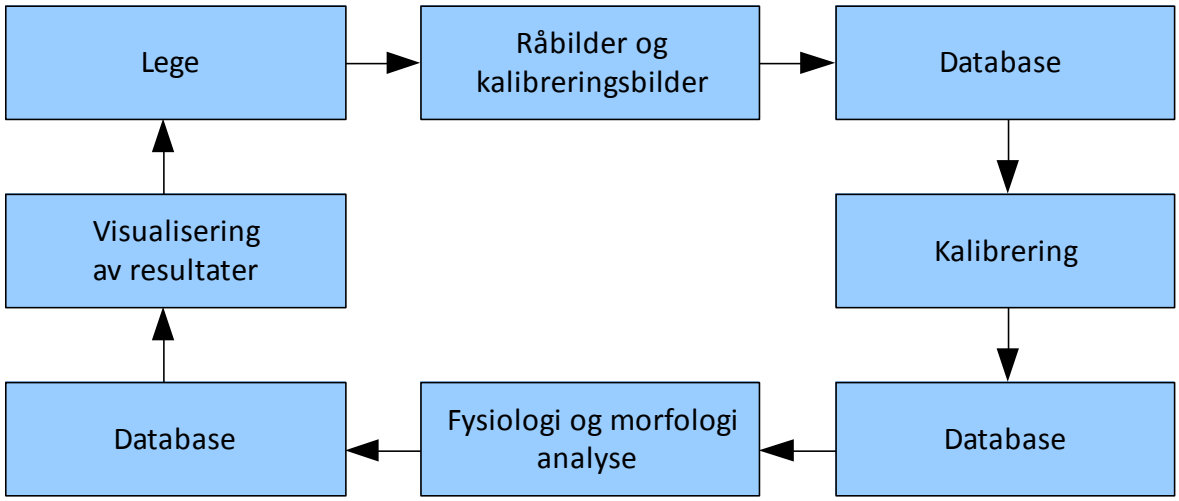
Samlet gir fysiologi og morfologi parameterne en god indikator på om lesjonen man undersøker er av type melanom eller ikke. Parameterne vil samlet sett gi legen eller spesialisten en tilbakemelding som er basert på mange variabler som er usynlig for det blotte øye.

4 Problembeskrivelse

4.1 Introduksjon

Balter Medical er en gründerbedrift der den grunnleggende algoritmeutviklingen foretas av fysikere. Fysikerne har ansvar for ulike deler av prosessen fra å ha ukalibrerte pasientbilder til forslag til en diagnose. For hver pasient vil det være lagret mange filer for hver level med data. Etter hvert som antall undersøkte lesjoner øker, vil datamengdene i mappestrukturen bli veldig omfattende.

4.2 Beskrivelse av diagnostiseringsprosess



Figur 4.1 Diagnostiseringsprosessen fra lege til resultat

Figur 4.1 viser prosessen fra lesjonsprøve hos legen til resultat. Legen tar bilder av lesjonen som han vil undersøke. Dette gir et resultat på 30 pasientbilder. På slutten av dagen vil legen også ta to sett med kalibreringsbilder. Den ene varianten av kalibreringsbildene blir tatt mot et helt hvitt motiv. Resultatet gir 300 CS bilder. Det andre kalibreringsbildesettet er bilder tatt av helt svart motiv, disse bildene kalles DC og er også 300 i antall. For å se eksempel på pasient-, CS- og DC – bilder se kapittel 7.1.4.

Samlingen av pasientbilder tatt av legen blir omtalt som både level 0 bilder og råbilder. CS- og DC-bildene blir, som nevnt over, omtalt som kalibreringsbilder. Råbildene og kalibreringsbildene blir sendt inn fra klinikkene til Balter Medical og lagt inn i databasen.

Kalibreringsprogrammet henter pasientbilder og kalibreringsbilder fra databasen. Kalibreringsrutinen kalibrerer level 0 bildene (se kapittel 3.1), resultatet kalles level 1 bilder legges i databasen.

Fysiologi og morfologi analysene henter de kalibrerte level 1 bildene og prosesserer dem. Resultatene herfra er ikke lenger bilder, men matriser og MATLAB – filer. I MATLAB – filene lagres det på nytt informasjon om lesjonen som finnes i metadatafilene. Metadatafilene er

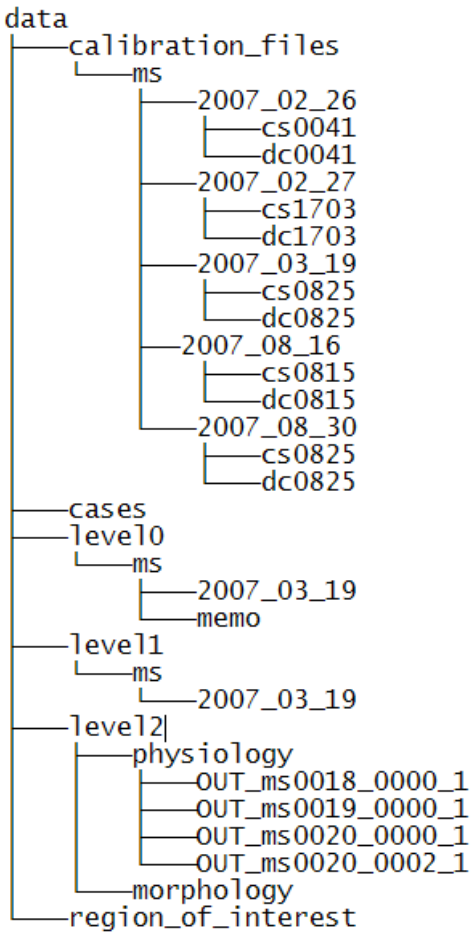
dagens metode for å lagre relasjoner mellom pasienter og plassering av data i filstruktur. Det lagres også mye annet som vil bli spesifisert i 6.1.2.

Det arbeides fremdeles med å forbedre algoritmene som produserer fysiologi- og morfologidata, så å lagre disse i databasen er ikke ønskelig på dette tidspunktet.

Data fra alle leveler kan vises med visualiseringsverktøy laget av en medstudent. Man kan dra slutninger ut fra visse egenskaper ved bildene når de har gått gjennom enten fysiologi- og morfologianalyse. Resultatene fra analysen sendes tilbake til klinikken og legen.

4.3 Dagens situasjon hos oppdragsgiver

All data hos Balter Medical blir med nåværende lagringsløsning lagret i filstruktur. Om det kan kalles en database er i beste fall diskutabelt. Figur 4.2 er et utsnitt av dagens mapper. Hovedinndelingen går på hvilken type data som er lagret i mappen. Deretter en forkortning som beskriver hvilket sykehus eller klinikk bildene er tatt, undermappene hvilken dato bildene er tatt.



Figur 4.2 Utskrift av mapper i eksisterende datastruktur

På kalibreringsbildene (calibration_files) vil mappen på nederste nivå fortelle om det er en samling med CS eller DC bilder etterfulgt av fire tall som beskriver om tidspunktet bildene er blitt tatt.

Level 0 og level 1 bilder er lagret i lik struktur. Hvilket level dataene er delt opp i som øverste mappe, deretter etterfulgt av en undermappe som representerer sykehuset eller klinikken bildene er blitt tatt, likt som kalibreringsdata. Mappen under sykehus eller klinikk representerer datoene det er tatt pasientbilder. Pasientbilder tatt på en respektiv dato vil ligge i mappen som representerer den datoen. Under hvert sykehus på level 0 vil det også være en memo - mappe. I denne mappen ligger metadatafilene som beskriver blant annet hvilke kalibreringsbilder som brukes til de ulike pasientdataene. I metadatafilene lagres også annen blant annet informasjon som klinisk- og patologisk diagnose, hvilken lege som har utført målingen og hvilken klinisk protokoll som er gjeldende under målingen.

Level 2 data er ikke strukturert like bra. Under fysiologi (fysiology) er det en mappe for hver av pasientene det er laget fysiologi data for. Under morfologi ligger det MATLAB filer som inneholder data om de ulike morfologiske faktorene beskrevet i kapittel 3.2.2.

4.4 Problemer med eksisterende løsning

Prosessen i dagens løsning involverer mange mennesker der de ulike har ansvar for sitt område. Man har et mål om at hele prosessen fra lege til resultat skal kunne automatiseres. For å oppnå dette er det viktig at man kan stole på at dataene på hvert nivå:

- Er lagret på det formatet man er blitt enig om.
- Har inkludert alle filer som skal være med og at filene ikke er tomme
- Ikke blir slettet ved en feiltagelse
- Er pålitelige, med andre ord at om man har sjekket at dataene finnes, så er de der når man skal bruke dem.

Dagens relasjoner mellom data går gjennom filnavn, mappenavn, metadatafiler og sist, men ikke minst ved hjelp av MATLAB – filer. I MATLAB – filene er det spørreskript som gjør rekursive kall i filstrukturen, og i noen tilfeller er det lange filer med hvis, eller hvis (if, else if) test. Erfaringsmessig sier brukerne av MATLAB skriptene at de gjør endringer dersom de ser noe de mener er feil. Dette kan gjøre utfallet av spørringen til noe brukeren ikke vil den skal gjøre i utgangspunktet.

Det er også et ønske om å frigjøre seg fra MATLAB, noe som gjør at mesteparten infrastrukturen de har for søking i data faller bort. Men man har fremdeles muligheten til å se gjennom filene i filstruktur. Da havner vi over på et annet problem. Et forsiktig estimat vil tilsi at man vil ha godt over 1 TB (terabyte) med bildedata før forsøkene er ferdig. Om man ser på Figur 4.2 er dette et lite utsnitt av det som allerede finnes i dag. Problemet er at også oversikteligheten i filstrukturen kan bli begrenset.

Tilgangsrettigheter er også et problem med dagens system. Ett brukernavn og passord brukes av alle brukere. Dette fører til at alle brukere har samme rettigheter på alle data. I fremtidig løsning er dette ikke ønskelig.

Som et lite grensetilfelle vil jeg ta med eventuell industriell spionasje. Tyveri av filer som ligger tilgjengelig i filstruktur, er enklere enn å stjele filer og data som er lagret i database.

4.5 Problemstilling

Med bakgrunn i problemene med dagens situasjon ønsket oppdragsgiver en lagringsløsning som gjør det enklere og raskere å kunne behandle dataene og relasjonene mellom dem. Den nye løsningen må ta høyde for at det kan bli lagt til nye level av data ved et senere tidspunkt, både fysiologi – og morfologi data, samt eventuelle nyutviklede level med data. Videre må det lages et innlastingsprogram for laste data til lagringsløsningen. Innlastingsprogrammet må kunne validere datasettene, med hensyn på størrelse på filer og om alle data er med i datasettet. Det skal lages klasser som skal hente ut kalibreringsbilder og pasientbilder til kalibreringsrutinen, og innsettingsmetoder som gjør at de kalibrerte dataene kan legges i databasen. En medstudent skal også hjelpes med spørringer for visualiseringsprogram.

5 Problemanalyse

Dette kapittelet skal formulere oppgaven nærmere, sammen med de ulike valg som er gjort for dem.

5.1 Delproblemer

Vil nå klargjøre de ulike problemstillingene beskrevet i forrige kapittel. Den viktigste delen er å lage en database for lagring av data. For å lage databasen må man identifisere de ulike datatypene som skal være med, identifisere hvilke data som kan knyttes til samme tabell og forholdet mellom tabellene. Neste problem er å lage et opplastingsprogram som laster opp data fra filstruktur til databasen. Siste del av oppgaven er å lage en klasse som kan brukes for å hente ut og sette inn data under kalibreringsprosessen av bildene. Selve kalibreringsprogrammet lages av en annen student parallelt med den praktiske delen av denne oppgaven.

5.2 Utviklingsmetode

Gjennom dette prosjektet er det ikke blitt fulgt noen utviklingsmetode slavisk. Den utviklingsmetoden som jeg har benyttet, ligger nærmest XP (Extreme Programming). XP har man som utgangspunkt at programkoden skal være nok dokumentasjon, jeg har derfor benyttet utfyllende variabelnavn i attributt-, prosedyre- og klassenavn. Ett annet prinsipp i XP er at utviklingen skal foregå i tett samarbeid med oppdragsgiver. Oppdragsgiver skal være tilgjengelig for spørsmål under hele utviklingen. I XP skal utviklingen skje ved hjelp av skriftlige brukerhistorier fra oppdragsgiver. I dette prosjektet har brukerhistoriene blitt overlevert muntlig.

Under utviklingen av databasen ble attributter, attributtnavn og entiteter etter mange interaksjoner godkjent av oppdragsgiveren. Også utviklingen av det grafiske grensesnittet var oppdragsgiver med for å gi kommentarer og godkjenning av design.

Hoveddelen av prosjektet har vært å lage en database som skal lagre data som nå blir lagret i filstruktur. Det har ikke vært lett å teste den i praksis før man har fått opplastingsprogrammet på plass. De ulike klassene som har i oppgave å kommunisere med databasen, ble sjekket hver for seg for å se at data som ble satt inn tilsvarte det man forventet. På opplastingsprogrammet ble hver av komponentene testet først i etterkant av oppdragsgiver etter hvert som de ulike grafiske komponentene var ferdig. Validering av at innsettingsmetodene har fungert har blitt vist med MySQL – spørringer som fant rette dataverdier i databasen. Visualiseringsverktøyet utviklet av en medstudent er også blitt brukt for å validere bildene som er blitt satt inn i databasen.

5.3 Kravspesifikasjon

Dette kapittelet tar for seg Balter Medicals tanker, ønsker og krav til systemet. Kravspesifikasjonen som presenteres er slik den fremsto mot slutten av oppgaven i henhold til XP utviklingsmetoden. Spesielt databasen ble mer omfattende enn det som ble skissert på de første møtene med oppdragsgiver, noe som også medførte at også opplastingsprogrammet ble mer omfattende. Opplastingsprogrammet fikk også heve kravene for validering av datasettene som skulle lastes opp.

Kravene deles inn i funksjonelle og ikke funksjonelle krav [9].

5.3.1 Funksjonelle krav

De funksjonelle kravene beskriver hvilke tjenester systemet skal levere og hvordan systemet skal oppføre seg og ikke oppføre seg i gitte sitasjoner.

Database

Data skal lagres i en relasjonsdatabase. Det skal være fokus på å få med alle relevante attributter som i nåværende system er lagret i metadatafiler, tekstfiler, MATLAB skript, mappe- og filnavn. Koblinger mellom data som finnes i metadatafilen skal også ivaretas i relasjonsdatabasen.

Saksidentifikasjonsnummer som er kombinasjonen av sykehus, pasientnummer, lesjon og måling skal knyttes til gyldige verdier av skanner, klinisk diagnose og klinisk protokoll. Sist men ikke minst skal ett saksidentifikasjonsnummer knyttes til de medisinske dataene som er lagret (level 0- og level 1-bilder) samt kalibreringsbildene som trengs for å kalibrere level 0 bilder til level 1 bilder (CS- og DC- bilder). Det skal legges opp til at man skal kunne legge til nye medisinske data (fysiologi og morfologi) i databasen, men dette er ikke en del av min oppgave og vil følgelig ikke bli beskrevet.

Sett med kalibreringsbilder skal kunne knyttes til mange saksidentifikasjonsnumre. Å lagre dem en gang for hvert saksidentifikasjonsnummer ville ført til store mengder med dobbellagring.

Klasser til kalibreringsprosessen for kommunikasjon med database

Det skal lages klasser som står for kommunikasjon med de ulike tabellene i databasen. Aktuelle innsettings- og uthentingskommunikasjon i de ulike klassene vil bli laget ved behov.

Det skal lages et objekt som kan brukes av kalibreringsrutinen , som blir laget av en annen student. Det vil være fokus på at objektet som lages skal kunne brukes etter hvert som databasen utvides med fysiologi og morfologidata.

Opplastingsapplikasjon

Opplastingsapplikasjonen skal være et program med GUI (grafisk brukergrensesnitt). For å kunne laste opp eller endre data i databasen må brukeren valideres av et påloggingsvindu der man kobler seg til databasen med brukernavn og passord.

Opplastingsapplikasjonen skal gi brukeren tilgang til grafiske vinduer der en kan legge til sykehus, kliniske protokoller, kliniske diagnoser, skannere og skannerinnstillinger. Blant skannerinnstillingene skal man kunne legge til nye kamerainnstillinger og lampeinnstillinger. For en kamerainnstilling skal man legge til relativt kalibreringsområde.

Applikasjonen skal kunne laste opp pasientdata som er beskrevet i en metadatafil. Metadatafilen og bildekatalogene den refererer til skal ligge i en forhåndsbestemt plassering i filstrukturen. Et problem når bildene blir sendt fra klinikkene til oppdragsgiver har vært at det mangler filer, eller at filene er tomme. Nåværende programvare til skannerne som brukes av allmennpraktiserende leger og dermatologer avhenger av at de selv skriver inn pasientnummer, lesjonsnummer og målingsnummer som lagres i metadatafilen. Feil i metadatafilen eller navnsetting av filnavn skal justeres av oppdragsgiver før man kjører opplastingsrutinen. Om det mangler filer, eller filene er tomme skal oppdages av opplastingsapplikasjonen. Under kjøring skal programmet gi tilbakemelding til brukeren om hvilke data som behandles, og om innsetting til database gikk bra eventuelt en feilmelding om det er noe som går galt. Tilbakemeldingen skal kunne lagres i tekstfil.

5.3.2 Ikke funksjonelle krav

Ikke funksjonelle krav beskriver begrensninger på de tjenester og funksjonene som er tilbudt av systemet.

Brukervennlighet

Brukergrensesnittet skal være intuitivt og lett å forstå av brukerne. Det forventes at brukere er kjent med attributtnavn, og andre terminologier fra nåværende system. Oppdragsgiver skal evaluere grensesnittet.

Språk

Nåværende system hos oppdragsiver benytter engelsk både i kode og variabelnavn. Oppdragsgiver driver utvikling i et internasjonalt arbeidsmiljø, og kreves at engelsk også skal bli brukt i den nye løsningen.

Programmeringsspråk

Oppdragsgiver ønsket at programvaren skulle skrives i C++.

Valget av grafisk bibliotek var ikke avgjørende for de grafiske oppgavene jeg skulle utføre. Kravet til det som skal vises grafisk er relativt begrenset. Som nevnt tidligere har vi vært 3 studenter som har jobbet med ulike deler av prosessen vist på Figur 4.1. En av studentene

jobbet med visualisering av data fra ulike level. Han bestemte seg for å benytte seg av biblioteket Qt [10]. Qt er et anerkjent grafisk bibliotek som kan brukes på flere operativsystem. Det viste seg også at Qt har gode klasser for databasekommunikasjon. Qt er et bibliotek som jeg har kjennskap til fra fag undervist på Høyskolen i Bergen. Disse punktene ble avgjørende for at også jeg valgte Qt.

For å velge databasespråk var prosessen noe lengre. Oppdragsgiver sendte meg på caBIG (cancer Biomedical Informatics Grid) [11] konferanse i Washington D.C.. caBIG er en organisasjon sponset av NCI (National Cancer Institute). NCI er en underdivisjon av NIH (National Health Institute) som er helsemyndighetene i USA. Formålet med turen var først og fremst å utforske mulighetene for at det fantes en databasestruktur som inneholdt mange av de samme parameterne som var aktuelle for oppdragsgiver og eventuelt tilpasse resten av databasestrukturen til det som er særegent for systemet de har. Eventuelle standardiseringer gjort på lagringsformat var for datadeling og lagring på sentraliserte servere. Det fantes ikke noen ferdig løsning som kunne tilpasses og implementeres hos oppdragsgiver. Fokuset ble da flyttet til om det var mulig å finne et spesielt databasespråk som ble brukt innad i caBIG. Spørsmålet kom opp på ett av seminarene, og svaret var at løsningen var bygget opp for å kunne være databaseuavhengig. Ikke overraskende med tanke på at caBIG har som mål at alle forskere og leger skal få dele sine data. Oracle Clinical ble nevnt som et databasesystem brukt av mange deltagere i caBIG strukturen. Oppdragsgiver ble presentert med forslaget, men var ikke interessert å i investere i lisensiert programvare. Forslaget som kom opp da var PostgreSQL og MySQL, som var åpne varianter jeg er kjent med fra Høyskolen i Bergen. Oppdragsgiver var kjent med bruk av MySQL fra forskningsarbeide tilknyttet CERN, så valget falt på dette databasespråket. MySQL [12] er en variant av SQL. SQL lagrer data som en relasjonsdatabase.

Operativsystem
Oppdragsgiver ønsket at utviklingen skulle gjøres i Linux. Som nevnt i punktet ”programmeringsspråk” kan man kjøre Qt i både Linux og Windows. Dette fører til at man kan kompilere koden uten store endringer for å utvikle en Windows variant av programmet. Fokuset i oppgaven min har vært å utvikle programmet i Linux – miljø. Linux distrubisjonen som ble benyttet i utviklingen er Ubuntu 7.1.

5.4 Verktøy

Under utviklingen av databasen er det viktig med tilbakemelding fra oppdragsgiver. For å tydeliggjøre relasjoner mellom de ulike tabeller og attributtnavn er det mer oversiktlig med bruk av UML (Unified Modeling Language) enn ren MySQL kode som beskriver tabellene. PowerDesigner er et verktøy som lar deg lage UML diagrammer og fysisk datamodell og MySQL kode. Har gode erfaringer med bruk av PowerDesigner på tidligere prosjekt, og å bruke dette her ble naturlig.

For å ha god oversikt med alle filer ble det besluttet å programmere i en grafisk IDE (Integrated Development Environment). Med bakgrunn av at biblioteket Qt ble valgt som bibliotek falt valget på utviklingsmiljøet QDevelop. QDevelop er en IDE som er utviklet for Qt 4.

Det ble brukt SVN (Subversion) som versjonskontroll system på filene mine. SVN ble satt opp på en ekstern server, så dette fungerte også som et sikkerhetskopisystem.

MySQL Query Browser [13] ble brukt for å utføre spørringer på databasen. Spesielt om spørringene inkluderer binære data har det vært greit å bruke MySQL Query Browseren. Vanlig terminalvindu taklet ikke å vise binærdata fra spørringene.

6 Løsningens oppbygning

Dette kapittelet tar for seg den nåværende løsningen og hvordan systemet er med den nye løsningen.

Nøyaktig hvilke bølgelengder, vinkler som benyttes for kamera og lamper, samt data knyttet til hvilke leger som har utført forsøkene regnes som forretningshemmeligheter og har derfor blitt endret i denne oppgaven. Dette er også nevnt tidligere i oppgaven.

6.1 Nåværende løsning

6.1.1 Skanner

Nåværende informasjon om skanneren lagres i en MATLAB fil kalt GetPtInfo som vises under. Pt er en forkortelse på prototype som er den gamle betegnelsen for skanner. En skanner identifiseres av 3 tall. Første tall beskriver skannernummer, neste vedlikeholdsnummer og det siste geometrisk kalibreringsnummer. Samlet refererer man til de tre tallene som skannervektor.

```
function p = GetPtInfo(SerialNumber)
% p = GetPtInfo(SerialNumber) returns information about geometrical and
% calibrational properties of the prototype with a given SerialNumber.
% Example:
% p = GetPtInfo([7 0 0])

p = [];
notFound = false;
ptVec = SerialNumber;
switch ptVec(1)
 case 7 %____________Prototype 7, ___________________________________________
  switch ptVec(2)
   case 0 % 7_0
    p.colors   = {'UVA'; 'violet'; 'blue';'cyan';'yellow';
                  'orange';'red';'red/NIR';'NIR1';'NIR2'};
    p.centWL   = [350 400 450 500 550 600 650 700 800 1000];
    p.lampAng  = [30 45 30 90 30 20 20 50 20 20];
    p.lampPhi  = [40 20 120 20 70 50 100 160 110 120];
    p.camAng   = [100 60 20];
    p.camPhi   = [120 20 22];
    p.BRDF_cal = BRDF_Mat(p.lampAng,p.lampPhi,p.camAng,p.camPhi);
    p.calFact  = 1./([1 1 1 1.5 1.5 1.5 2 2 2 2]*pi.*65535);
    switch ptVec(3)
     case 0 % 7_0_0
      p.relcal_area.y(1,:)={[100:160]};
      p.relcal_area.x(1,:)={[710:760]};
      p.relcal_area.y(2,:)={[120:172]};
      p.relcal_area.x(2,:)={[698:776]};
      p.relcal_area.y(3,:)={[198:237]};
      p.relcal_area.x(3,:)={[701:768]};
    end
   otherwise
    notFound = true;
  end
 otherwise
  notFound = true;
end
```

Selv om bølgelengder, kamerainnstillinger og lampeinnstillinger er parametere som er like for de fleste skannere, lagres de på nytt for hvert nytt skannernummer man lager. Informasjonen om bølgelengder, lampe- og kamera vinkler er viktig for kalibreringsprosessen på level 0 data. Lagringsstrukturen er lett å lese, men siden nesten all informasjon som lagres for ett skannernummer brukes igjen på den neste, gir dette mye dobbellagring av informasjon. En annen faktor som bør tas hensyn til er at man over tid vil få lange en lang fil. Oversiktlighet i selve strukturen krever at nye elementer som vil bli lagt til er sortert etter de tre tallene i skannervektoren.

På selve skanneren er det en liten tapp klistret på innsiden av glassplaten. Området kalles relativt kalibreringsområde (Relcal area). Dette området brukes i en rutine som ser på tap av lysintensitet på lampene over tid.

6.1.2 Metadata

Dette er en omfattende fil som knytter sammen alle pasientdata. Under vises en linje fra en metadatafil. Feltene blir skilt med semikolon.

```
MS IRB 2006;CIP 1.0;[6 1 1];dr. Doolittle; 2007_08_30;14:04;ms0057_0002;
  2007_08_30/cs0825; 2007_08_16/dc0815; 3 ;left neck;m1960;atypical
  nevus, 0 room 08/ category 3;biopsy_yes; compound nevus; with
  architectural disorder;
```

De to første punktene beskriver hvilken klinisk protokoll som var gjeldende når forsøket ble tatt. Neste punkt beskriver skanneren som ble brukt i forsøket. Deretter kommer legen som foretar målingen. Dette feltet fylles manuelt av legene, så ved enkelte tilfeller skriver de ned alt medisinsk personell som var til stede når målingen ble tatt. Neste felt beskriver datoen målingen ble tatt, etterfulgt klokkeslett målingen ble tatt.

Punktet som kommer nå er det viktigste, hvilken pasient dataene tilhører. Av personvernshensyn lagres ikke navn, personnummer eller noe som lett kan identifisere hvilken pasient målingen er tatt av. Dette er heller ikke viktig for diagnostiseringen. Men ms0057_0002 illustrerer ikke bare hvilken pasient bildene representerer, den beskriver hvilke lesjoner man undersøker og hvilken måling undersøkelsen er av den gitte lesjonen. Samlet refererer man til det som ”Case id” eller saksidentifikasjonsnummer på norsk.

- ms – forkortelse på sykehusnavn, tilsvarende det man kan finne i filstruktur.
- 0057 – pasientnummer. De fire første tallene i saksidentifikasjonsnummeret beskriver pasientnummer. Pasientnumrene tildeles inkrementelt etter hvert som man behandler nye pasienter.
- 00 – lesjonsnummer. De to første tallene i den siste gruppen på fire. Starter tellingen på 0 og inkrementeres etter hvert som man undersøker nye lesjoner.
- 02 – målingsnummer. De to siste tallene i den siste gruppen på fire. Målingsnummeret brukes fordi man kan ønske å ta flere målinger for samme lesjon. Også denne starter tellingen på 0, så i dette tilfelle er det måling nummer 3.

Neste punkt beskriver stien til kalibreringsbildene som skal brukes for å kalibrere level 0 pasientbildene. DC bildene blir tatt samtidig som CS bildene, og lagres under samme dato og tid. For å hente ut dem bytter man bytter bare ut ”cs” med ”dc”. Feltet etter beskriver et eventuelt DC2 kalibreringssett. Siste feltene beskriver kategorien for klinisk diagnose, lesjonens plassering, kjønn og fødselsår på pasienten, kommentarfelt for dermatologene, om det ble tatt biopsi, patologisk diagnose og kommentar til den patologiske diagnosen.

Denne måten å knytte dataene sammen på har fungert greit til spørring. MATLAB skriptene leser feltene fra metadatafilen og utfører spørringer ut fra dataene den trenger. Det er en lite effektiv måte sammenlignet med metodene en kan bruke eksempel med en relasjonsdatabase. Metadatafilene er meget uryddige å lese. Hver linje i en metadatafil inneholder 16 felter. Dette gjør at det er vanskelig å få oversikt over et saksidentifikasjonsnummer ved å lese i filen. Hver fil inneholder mange linjer. Dette bidrar også til å begrense oversikten. Om man ved en feiltagelse skulle være uheldig å endre verdier på steder man ikke hadde planlagt kan det være vanskelig å lokalisere feilen i etterkant.

6.1.3 Filstruktur

På Figur 4.2 som illustreres tidligere i oppgaven viser filstruktur med mapper. Nå vil jeg gå inn på selve datafilene som ligger i filstrukturen. Forrige delkapittel beskrev metadatafilen som knytter sammen relasjonene mellom pasient, diagnoser og så videre. Dette delkapittelet skal gå mer inn på selve bildene som brukes i bildediagnostiseringen.

Først en liten repetisjon fra mappestrukturen. Først hvilket datanivå dataene er fra. Så sykehuset de er tatt på, deretter datoen de er blitt tatt. Disse parameterne samsvarer med dem man finner i metadatafilen.

Kalibreringsfiler og pasientfiler har en del felles trekk. I kapittel 4.2 ble det beskrevet at for pasientbilder ble det tatt ett bilde med hvert kamera for hver bølgelengde, mens for kalibreringsbilder blir det tatt 10 bilder for hvert kamera for hver bølgelengde. Informasjon om hvilket kamera som ble brukt, og hvilken bølgelengde lesjonen ble belyst med er viktige faktorer til videre bruk. Dette er lagret i filnavnet både hos kalibrerings - og pasientbildene.

Eksempler på filsti og filnavn:

Pasientdata:
data/level0/ms/2007_08_30/ms0057_0002_view1_350nm_200ms.tif

Kalibreringsdata:
data/calibration_files/ms/2007_08_30/cs0825/0825_cs0_view3_350nm_220ms.tif

Dette er et eksempel på filsti og filnavn på et eksisterende pasientbilde av typen level 0 og kalibreringsbilde av typen CS i dag. Vil først se på de datafeltene i filnavnet som er felles for begge varianter. Om man ser på eksemplene, så representerer tallet etter ”view” hvilket kamera som er brukt, ”350nm” er bølgelengden som er brukt for å ta bildet. Siste parameter

de har til felles er eksponeringstiden. Det er siste parameteren som står i filnavnet før filetternavnet, eksempelvis 200ms for pasientdataeksempelet.

Dette var de delene som var felles, så en gjennomgang av de delene som er spesifikke for de ulike variantene.

Først en gjennomgang av pasientdata. Her representerer første delen av filnavnet pasientdataene tilhører saksidentifikasjonsnummeret som ble presentert i forrige delkapittel.

For kalibreringsdataene vil jeg også nevne mappen filene ligger i. ”cs” representerer at det er ett sett med CS – bilder, de fire tallene i mappenavnet beskriver tidspunktet kalibreringsdataene ble tatt. Når en beveger oss over til filnavnet til kalibreringsbildet ser man at man finner igjen klokkeslettet i filnavnet. Siste feltet som da ikke er forklart er ”cs0”. ”cs” representerer at dette er en CS - fil. For hvert sett med CS- og DC - bilder blir det tatt 10 bilder med hvert kamera for hver bølgelengde. Tallet representerer hvilket nummer det er i sekvensen. Tidlig i oppgaven var inntrykket at det ikke var viktig å ta vare på sekvensnummeret i databasen, men det viste seg at de første bildene i bildeserien (alle bilder tatt i serie 0) var av dårlig kvalitet, og var derfor ønskelig å kunne identifisere disse og utelukke dem fra kalibreringsrutinen. Prinsippet å ta vare på alle data etter ønske fra oppdragsgiver gjorde at også disse lagres i databasen, til tross fra at de i praksis ikke vil brukes.

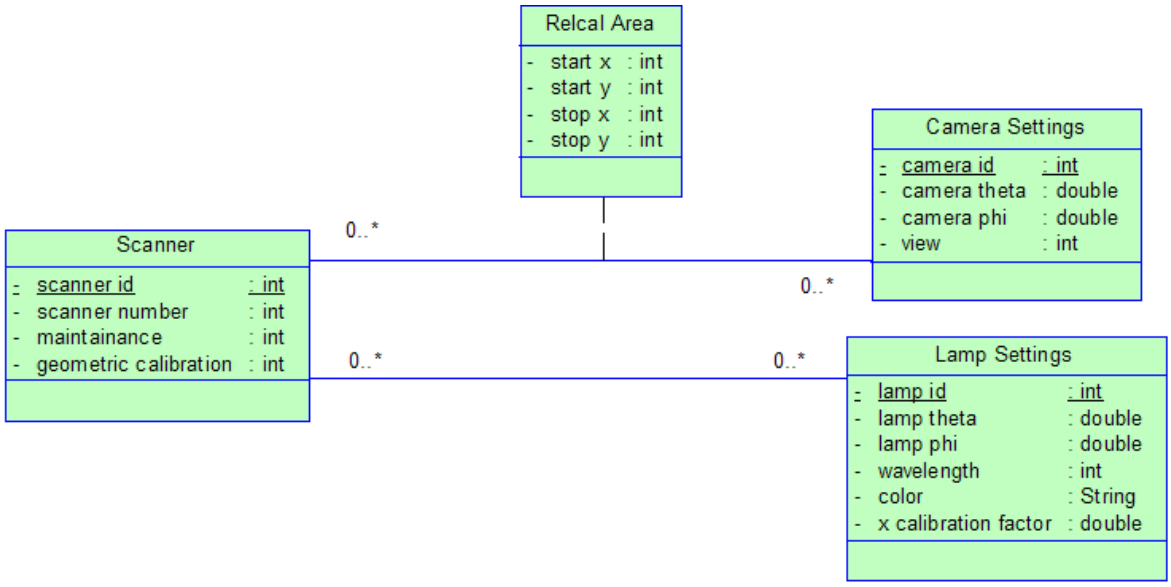
6.1.4 Kalibreringsprosessen

Dagens metode for kalibrering tar en liste ned saksidentifikasjonsnummer som input. For hvert saksidentifikasjonsnummer blir det gjort et oppslag i metadatafilen for å identifisere kalibreringssettene den er tilknyttet. Kalibreringsprogrammet vil da hente filene den trenger ved å traversere gjennom filstrukturen. Man kjører spørring på MATLAB skriptet GetPtInfo for å hente ut dataene som er knyttet til skanneren. Denne metoden er omtalt i Skanner tidligere i kapitlet.

6.2 Ny løsning

Den nye løsningen er bygget opp med en MySQL database for lagring og C++ klasser for håndtering av data inn og ut fra databasen. Den nye løsningen trenger også et program som kan brukes til opplasting av data fra filstruktur til database. Videre i dette delkapittelet vil jeg vil vise UML – diagram for hvordan dataene vil bli lagret i databasen og vise klassediagram over hvordan strukturen vil være på opplastingsprogrammet.

6.2.1 Skanner



Figur 6.1 UML diagram av skanner

Som man ser på Figur 6.1 er skannervektoren en tabell for seg selv. Kombinasjonen av skanner nummer, vedlikeholdsnummer og geometrisk kalibreringsnummer er i seg selv egentlig unikt, men da dette ville gitt veldig mange parametere som fremmednøkler når skanneren skulle kobles med andre tabeller i databasen valgte jeg å ha en primærnøkkel uavhengig av disse. Kamera – og lampeinnstillinger er blitt egne tabeller, slik at oppføringene bare vil bli lagret en gang selv om de brukes av mange skannere.

Relativt kalibreringsområde gir koordinatene til en tapp på glassplaten i skanneren. Disse koordinatene vil avvike litt for hvert kamera og må derfor lagres spesifikt for hvert kamera. Tappen blir ikke plassert på noen bestemt plassering for alle skannere, så disse koordinatene kan ikke lagres direkte i kamerainnstillingstabellen. Koordinatene knyttes derfor til koblingen mellom skanner og kamera innstilling.

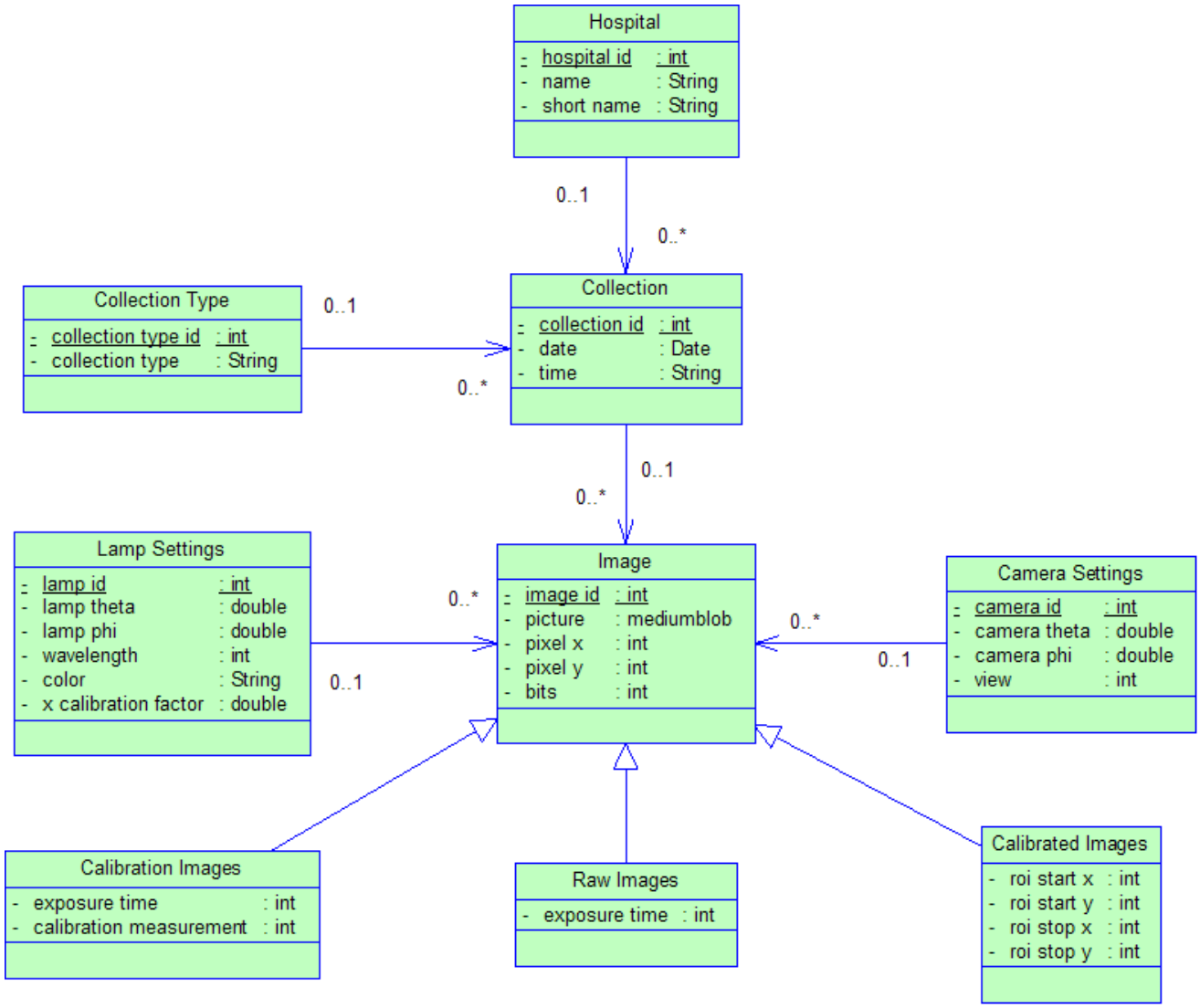
Som tidligere nevnt er det viktig å ta vare på alle variabler som ble lagret i filstruktur og MATLAB – skript i den nye databaseløsningen. Det er gjort noen justeringer i databasesystemet i forhold til det som ble lagret i MATLAB – skriptet GetPtInfo. Som nevnt over refereres verktøyet legen bruker nå som skanner. For lampe og kamera refereres det

som tidligere ble kalt vinkel (angle) som theta, dette er mer beskrivende på hvordan vinkelen er i forhold til planet. En annen justering er at man i databasen lagrer variablene ”x kalibreringsfaktor” istedenfor ”kalibreringsfaktor”. Grunnen til at jeg har hermetegn rundt dem er at man ikke skal tro at ”x kalibreringsfaktor” er det samme som variabelen x multiplisert med ”kalibreringsfaktor”. Tvert i mot kan forholdet mellom dem beskrives med formelen under.

$$\frac{1}{"x\ kalibreringsfaktor"\ *\ \pi * 65535} = "kalibreringsfaktor"$$

X kalibreringsfaktoren ligger også inne som en tabell i skriptet under variabelen ”calFact”. Dette er variabler med et lavt heltall og en desimal presisjon. Kalibreringsfaktoren derimot er et tall med mange desimaler. Som man kan se vil skriptet kalkulere kalibreringsfaktoren ut fra formelen over. Databasen legger opp til at man skal bruke x kalibreringsfaktoren og kalkulere kalibreringsfaktoren selv.

6.2.2 Samling med bildediagnostiske data og kalibreringsbilder



Figur 6.2 UML diagram av bildediagnostiske data

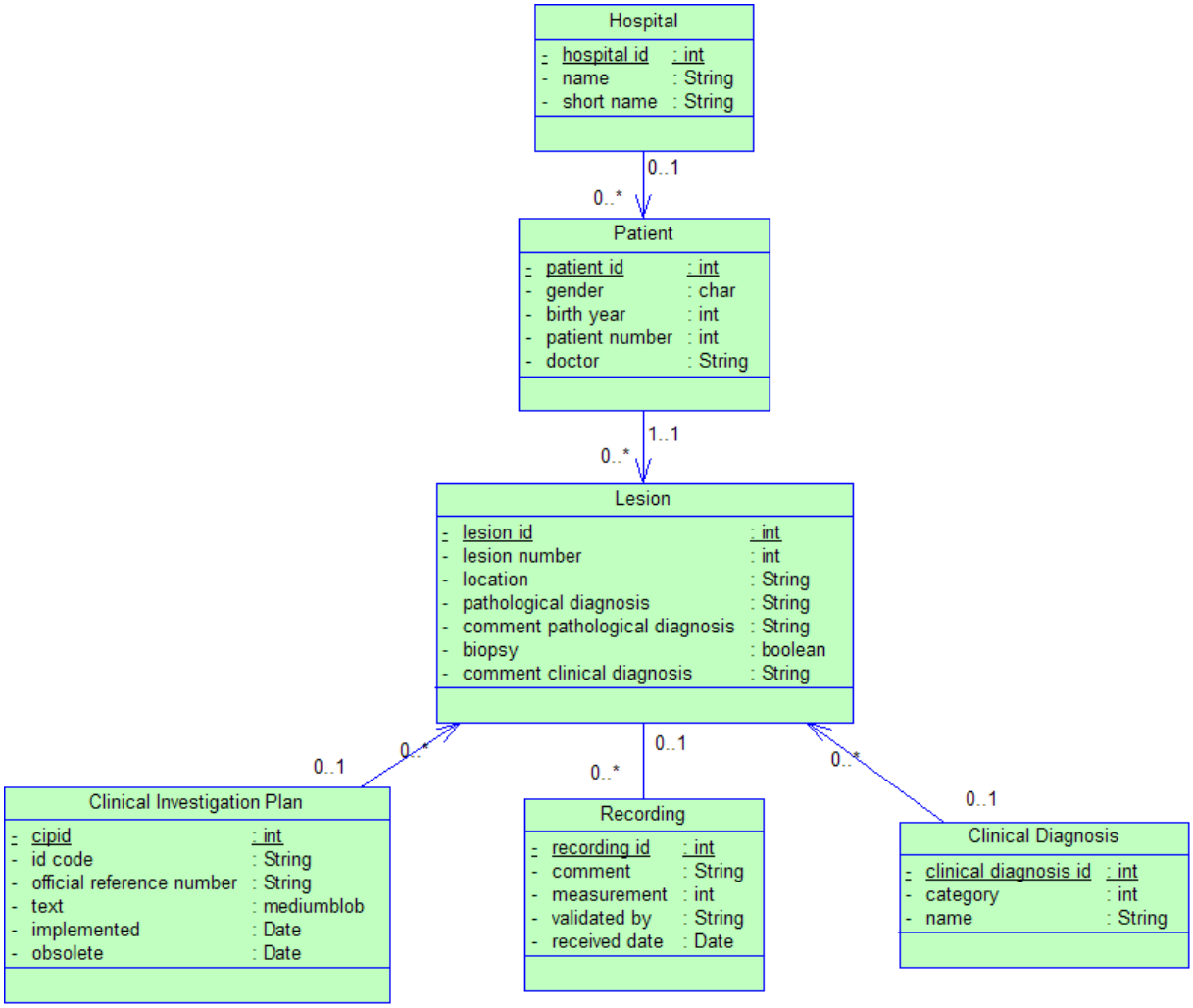
Både kalibreringsdata og pasientbilder har mange like egenskaper. De er begge en samling med flere bilder, og har begge både lampe og kamerainnstillinger. Løsningen for å samle mange bilder som en samlet gruppe ble å lage en samlingstabell (Collection). En annen tabell samlingstype (Collection Type) får ansvaret på å holde rede på hvilken type samling man har. Samlinger som støttes med nåværende versjon av databasen er ”calibration standard”-, ”dark current”-, ”calibration matrix”-, ”level 0”- og ”level 1” - samlinger. For bilder av alle samlingstyper er det viktig å vite hvilke kamera – og lampeinnstillinger som er brukt når bildet ble tatt. Innstillingene brukes når bildene skal kalibreres og når de skal brukes i fysiologi- og morfologianalyse.

Arv kan brukes for å kunne samle de attributtene som er felles for flere tabeller i en felles tabell. Siden alle bildetypene har mange like attributter var det naturlig å bruke arv. De ulike bildetypene har noen attributter som ikke passer for en felles tabell. For level 0- og CS- og

DC- bilder er eksponeringstiden viktig. CS og DC- bildene har også eksponeringstid, men som nevnt over er det også viktig å holde orden på hvilket nummer i sekvensen bildet er tatt, og det ble derfor naturlig å skille CS- og DC bilder fra level 0 bilder. For de kalibrerte bildene lagres det koordinater som beskriver hvor lesjonen er på bildet. Dette er "roi" koordinatene som lagres i tabellen kalibrerte bilder (Calibrated Images), hvor "roi" er en forkortelse av det engelske "region of interest".

Om man i fremtiden vil lagre fysiologi og morfologidata, som ikke er samlinger av bilder, vil det være naturlig å legge på et ektra felt i samlingstypen (Collection Type) som beskriver om samlingen er en samling med bildedata, fysiologi- eller morfologi data.

6.2.3 Pasientdata som tidligere var omtalt i metadata



Figur 6.3 UML diagram av lagring av data knyttet til pasient

Tidligere løsning var et system som reelt sett ikke var designet med hensyn på pasienter, lesjoner og målinger. Saksidentifikasjonsnumrene spesifisert i metadatafilen kan deles opp og grupperes i etterkant, dersom man vil gruppere dataene med tanke på pasienter og lesjoner. Men denne oppdelingen av data bør være en unødvendig prosess. Den nye

løsingen vist med UML på Figur 6.3 retter på dette. For hvert sykehus lagres det pasienter, identifisert utad med pasientnummeret. For hver pasient lagrer man lesjonene man undersøker og for hver lesjon lagrer man hver av målingene (Recording) man tar. Denne strukturen hindrer at man i verste fall lagrer forskjellig diagnose for samme lesjon, eller får andre ukonsistente data, noe som kan bli tilfellet når man gjør et søk i metadata.

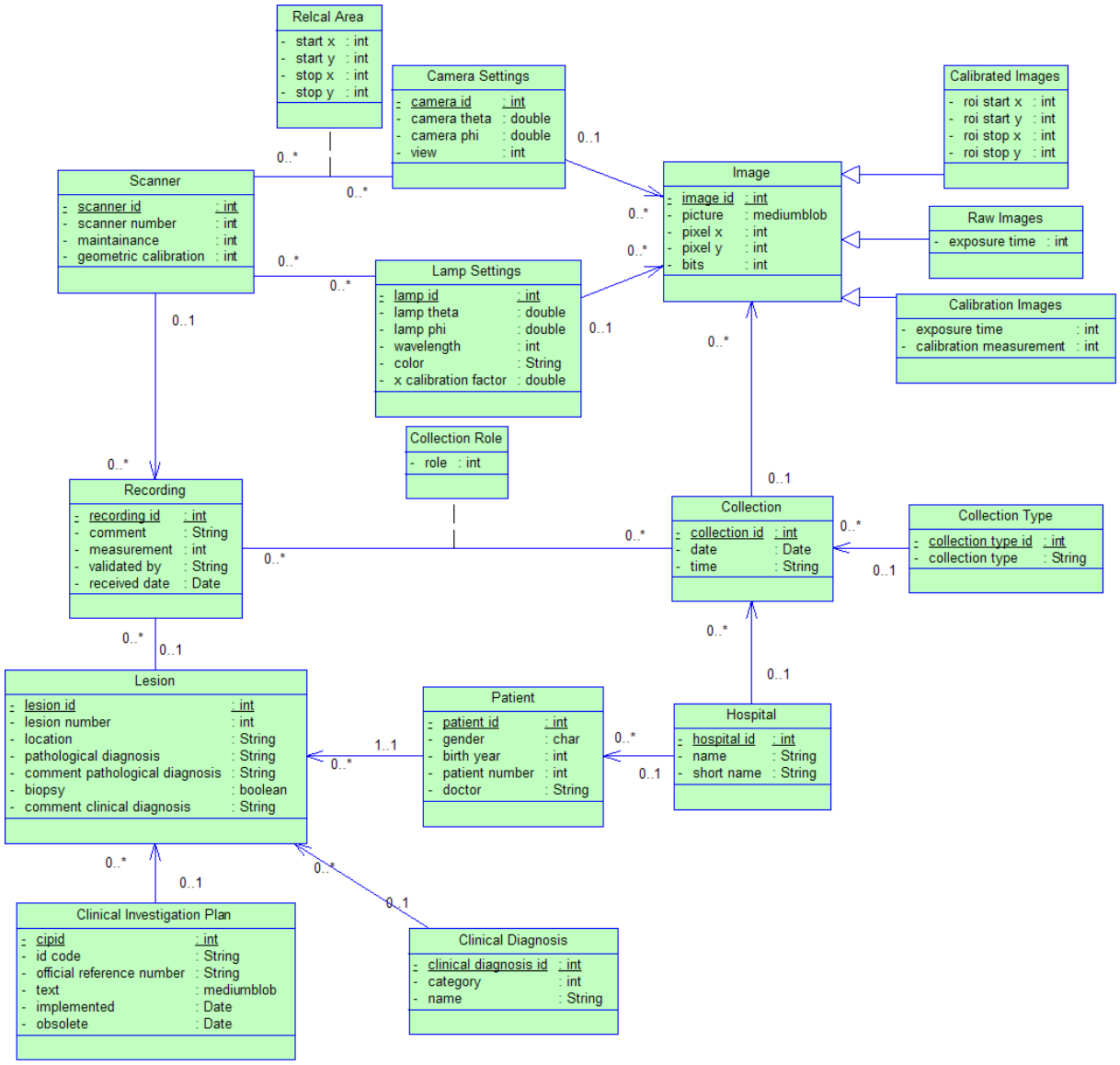
Det er viktig å holde orden på hvilken klinisk protokoll (Clinical Investigation Plan) som er gjeldende når man gjør et forsøk. Grunnen til dette er at koblingen til den kliniske protokollen knyttes til lesjon og ikke målinger eller pasient, er relativ enkel. Som nevnt tidligere i oppgaven har man av personvernhensyn ikke lov å lagre navn på pasienten. I dagens programvare til skanneren skriver legen selv inn pasientnummer, lesjonsnummer og målingsnummer. Dette vil i praksis si at det i mange tilfeller er lettere å gi pasienten et nytt pasientnummer for hver gang han eller hun skal undersøkes.

Løsningen min kan ikke ta som utgangspunkt at dette er tilfelle. Jeg har forutsatt at over tid er det vanskelig å huske hvilken føflekk som er blitt tildelt et spesifikt lesjonsnummer. Om man ønsker å undersøke den samme føflekken flere ganger i et spredt tidsrom, kan man ikke ta sjansen på at man husker hvilket lesjonsnummer den ble tildelt. Et eksempel er om skal undersøke tre føflekker på en pasient. De nummereres fra 0 til 2. Et par dager senere finner legen ut at føflekkene skal skannes på nytt. Det er vanskelig å huske hvilken føflekk som ble tildelt lesjonsnummeret 1. Det er derfor rimelig å anta at føflekkenes lesjonsnummer, på den nye runden med skanning, vil nummereres fra 3.

Et annet moment er at klinisk protokoll sjeldent endres. Problematikk med at feil klinisk protokoll knyttes til et lesjonsnummer er derfor høyst usannsynlig.

En annen ting som kanskje overrasker, er at man ikke lagrer lege som en egen tabell. Som med pasient-, lesjons- og målingsnummer skrives legens navn inn av legen selv. Etter å ha sett gjennom metadatafilen ble det fort klart at dette ikke er et standardisert felt. I enkelte tilfeller vil legen skrive ned navn på alle som var til stede når målingen ble tatt. Siden navn på lege ikke er standardisert og man lett kan søke på navn innad i et tekstfelt falt valget på å lagre informasjon om legen som et attributt og ikke som en egen tabell.

6.2.4 Oversikt over hele databasen



Figur 6.4 UML diagram over hele det nye databasesystemet

Figur 6.4 viser de ulike delene av databasen satt sammen til det nye databasesystemet. Den eneste tabellen som ikke er presentert til nå i det nye databasesystemet er, samlingsrolle tabellen (Collection Role). Denne tabellen er med for å kunne skille DC2 samlingen fra DC samlingen i relasjonen mellom måling (Recording) og samling (Collection). Som nevnt tidligere er DC2 en DC samling som bare opptrer med en spesiell rolle for utvalgte pasienter. Løsningen ble så enkelt som at man skiller dem på en heltallsverdi, verdien 1 beskriver at samlingen har rolle som en DC2 samling, 0 beskriver at samlingstypen som er spesifisert ved en relasjon mellom samling og samlingsrolle er riktig.

Skannertabellen blir knyttet til målingstabellen (Recording). Det var en grundig evaluering på om koblingen fra skannertabellen skulle knyttes til målingstabellen, eller om den skulle

knyttes til samlingstabellen (Collection). Den ene grunnen for at det endte opp med å knyttes til målingstabellen er at man i metadatafilen bare får oppgitt skannervektoren som knyttes til saksidentifikasjonsnummeret. Man kan egentlig bare anta at den samme skanneren er blitt brukt til CS- og DC – bildene som er knyttet til pasientbildene. Videre vil det aldri bli registrert en måling uten at man har tatt bilder med en skanner, så å knytte skanner til målingstabellen er det det samme som å ha skanneren knyttet til level 0 samlingen i dagens database. En annen grunn til at jeg valgte å koble den til målingstabellen er med tanke på fremtidlig utvikling av systemet. Dagens løsning legger opp til at man vil ta vare på alle data. Om en i fremtiden bestemmer seg for ikke å ta vare på level 0 data, og bestemmer seg for at det er nok med level 1 data, kunne man i verste fall risikere at koblingen mot hvilken skanner som er blitt brukt vil gå tapt, eller at man må modifisere databasen, spørringer eller funksjoner i programmene som bruker databasen.

6.2.5 Opplastingsprogram fra filstruktur til database

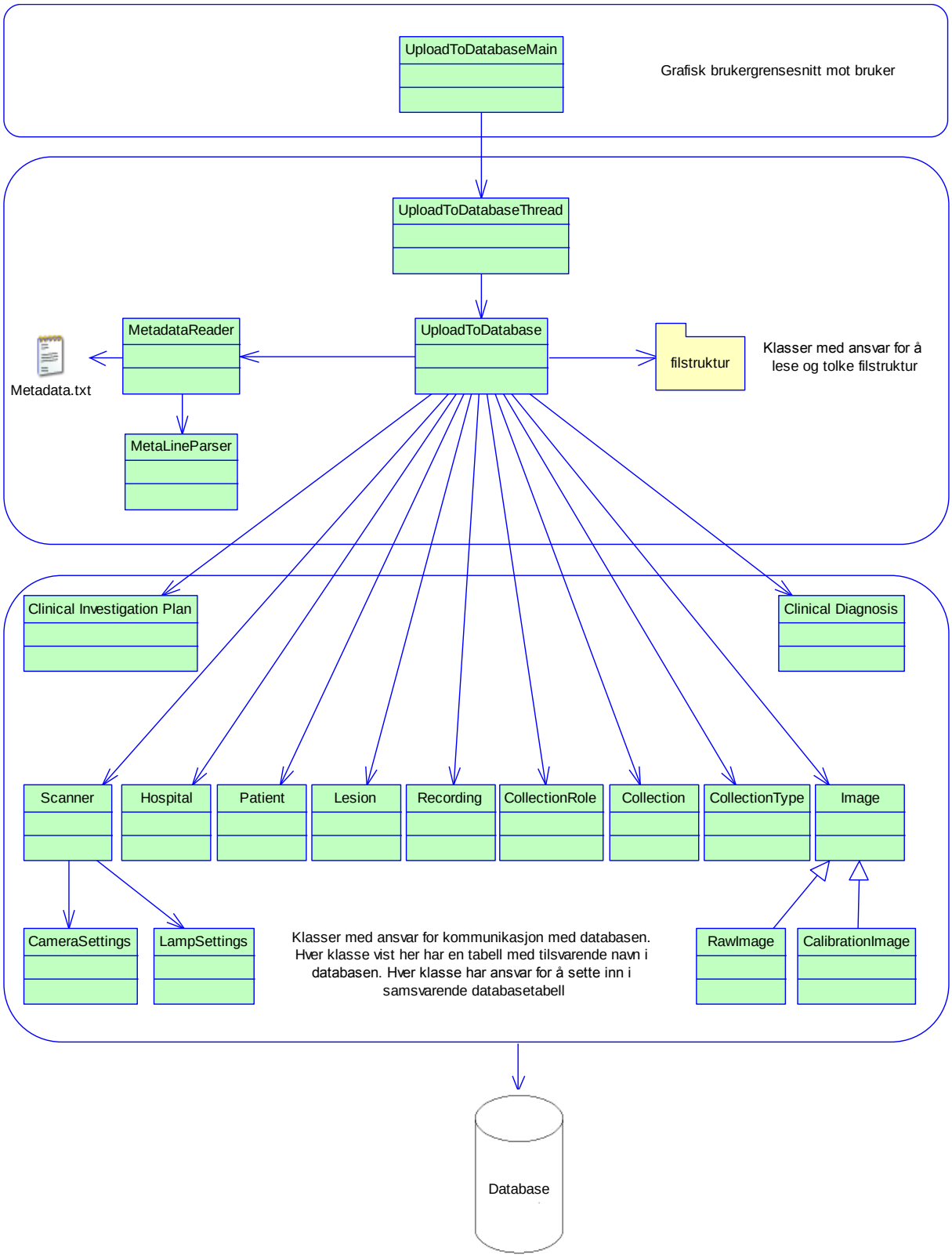
Opplastingsprogrammet har som hovedoppgave å få lastet opp pasient- og kalibreringsdata fra filstruktur og koble dem sammen slik at ingen data som finnes i metadatafilen går tapt. Men man skal også kunne betjene de delene som ikke er pasient og kalibreringsdata. Da tenker jeg på skanner, kobling mellom skanner, lampe- og kamerainnstillinger, kliniske protokoller og klinisk diagnose osv. Alt dette må kunne gjøres i et grafisk grensesnitt.

Innlesing ved hjelp av metadatafilen er en automatisert prosess, der det eneste som kreves av brukeren er at filene er tilgjengelig på strukturen som er avtalt, at feltene i metadatafilen er riktig, og at det allerede ligger oppføringer på innstillinger og lignende som beskrevet over. Strukturen for denne delen er vist ved hjelp av et klassediagram på Figur 6.5 under. Etter første kjøring av programmet viste det seg at det grafiske grensesnittet låste seg mens opplastingsrutinen kjørte. Dette ble løst ved å kjøre opplastingsrutinen som en tråd. Tråden har ansvar for å kjøre opplastingsrutinen, samt gi tilbakemelding til hovedvinduet om progresjon i programmet.

Metadatafilen som ble brukt tidligere hadde færre felt enn den nåværende versjonen. Noe av informasjonen ble lagret andre steder. Oppdragsgiver oppdaterte metadatafilen slik at opplastingsprogrammet ikke skulle måtte analysere flere tekstfiler for å få tak i all relevant til et saksidentifikasjonsnummer. For å ta hensyn til eventuelt fremtidige utvidelser av metadatafilen ble tolkeren av metadata skilt ut som en egen klasse. Metadata – leseren (MetadataReader) leser en linje fra metadatafilen og sender den videre til tolking i meta – linje analyseklasse (MetaLineParser). Den har som ansvar å lese hvert felt i linjen og oversette fra tekst til riktig datatype.

Opplastingsprogrammet (UploadToDatabase) benytter seg av metadata – leseren og validerer kalibrerings – og pasientdataene før de lastes opp til database. Opplastingsprogrammet benytter seg av metadata – informasjonen, samt informasjonen man finner i filnavnet og oppretter objekter. Objektene som har ansvar for kommunikasjon

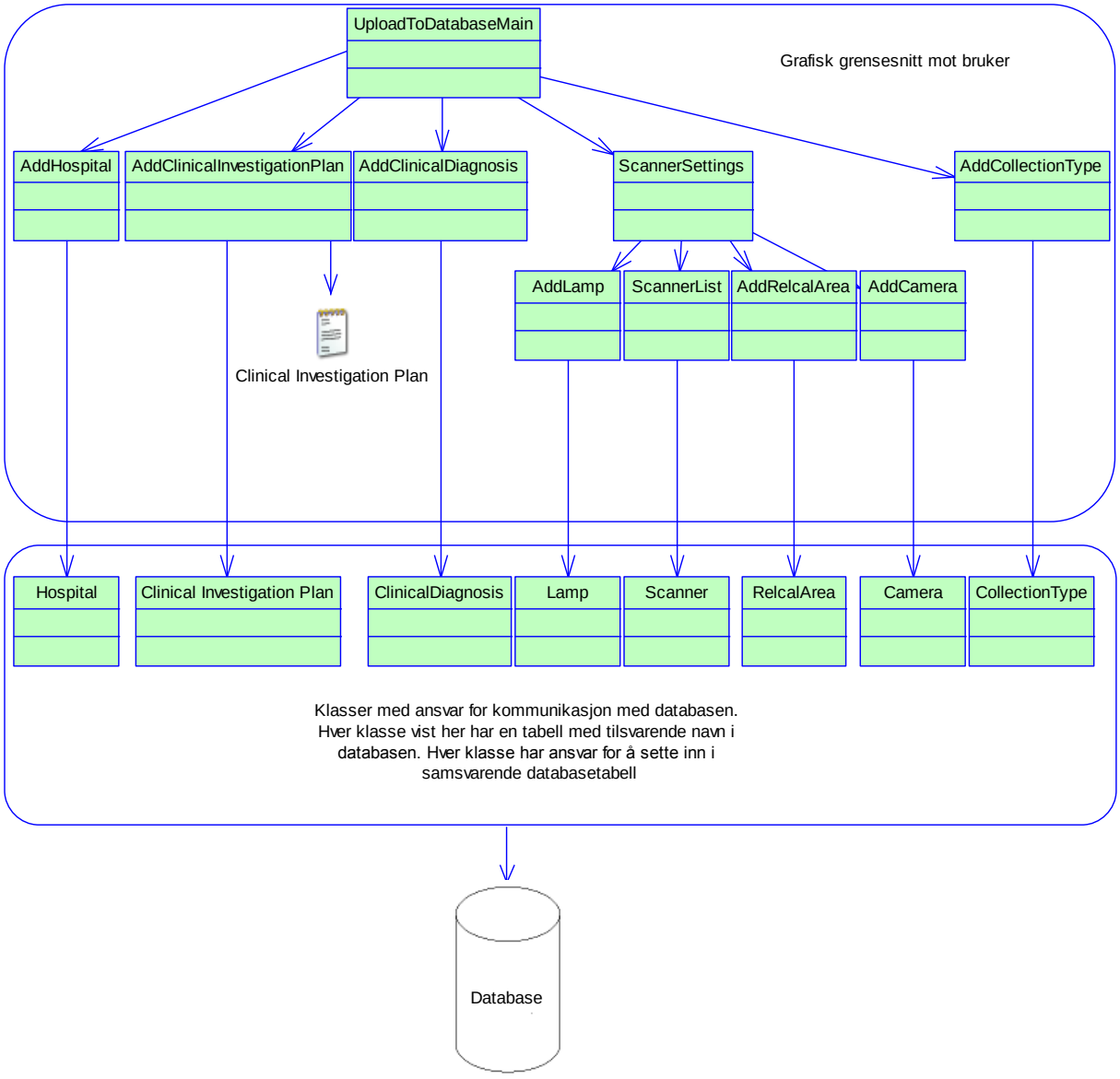
med databasen har som nevnt tidligere en tabell i databasen som har tilsvarende datafelt og datatyper. Databaseobjektene sørger for innsetting i databasen.



Figur 6.5 Klassediagram av den delen av opplastingsprogrammet som laster opp data som er beskrevet i metadatafil

For at bildedataene skal legges inn er det en betingelse at skanneren som er brukt må finnes i databasen. Dette er for at man ikke skal kunne ende opp med at data som ligger i databasen, blir identifisert med en skanner som har 10 gitte bølgelengder, mens bildene som man påstår har blitt tatt med samme skanneren bruker andre bølgelengder. Dette ville laget vanskeligheter for kalibreringsrutinen og videre analyse.

For at pasientdata skal lastes til databasen må man også ha lagt inn den kliniske protokollen og kliniske diagnosen. Denne delen av programmet styres av brukeren, og er mindre automatisert. Her skal brukeren selv skrive inn data. Eneste sjekk man får er at man ikke har ugyldige verdier, eller mangler data i innskrivingsfelt.



Figur 6.6 Klassediagram av den delen av opplastingsprogrammet som laster opp data angitt av brukerne

Figur 6.6 viser klassediagram av den brukerstyrte delen av programmet. Tankegangen er mye den samme som på den automatiserte delen. Forskjellen er at i denne delen av programmet kommer informasjonen fra bruker, ikke fra filstruktur. Unntaket er når man skal

laste opp klinisk protokoll (Clinical Investigation Plan), da spesifiserer bruker filsti til det offisielle dokumentet der den kliniske protokollen blir beskrevet.

Et litt spesielt tilfelle finner man også i skannerinnstillinger (ScannerSettings). Denne delen av programmet lar brukeren opprette nye skannere, kamera- og lampeinnstillinger. Man får muligheten til å legge til og fjerne kamera- og lampeinnstillinger til en valgt skanner. Denne metoden henter data som allerede finnes i databasen og tilbyr disse når man skal legge til eller fjerne innstillinger for valgt skanner. Når man legger til en kamerainnstilling må man også spesifisere det relative kalibreringsområdet. Om kalibreringsområdet viser seg å være feil, vil man kunne endre dette i ettertid.

6.2.6 Kalibreringsprosessen

Til bruk i kalibreringsprosessen ble det laget en klasse som ble kalt Case Id. Den nye klassen har en statisk metode som henter ut en liste med alle saksidentifikasjonsnumre som ikke er kalibrert. Rutinen oppretter et objekt av typen Case Id som representerer saksidentifikasjonsnummeret man ønsker å kalibrere. Det nye objektet henter ut rett skanner som passer til saksidentifikasjonsnummeret. Fra skannerklassen kan man finne de kamera- og lampeinnstillingene som gjelder for bildene man skal se på. Rutinen begynner med å se om kalibreringsmatrisen allerede er laget. Hvis den ikke er det, henter den ut CS- og DC – bilder for gitte bølgelengder. Kalibreringsmatrise er matrise med matriser utledet av CS- og DC- bildene knyttet til en måling. Case id objektet har muligheten til å legge inn oppføringer av type kalibreringsmatriser i samlingstabellen. Dette sparer tid for kalibrering av level 0 pasientbilder som bruker samme CS – og DC - bilder. Om nødvendig kan også DC2 bildene hentes ut av databasen. Case id objektet lar kalibreringsrutinen hente ut ett og ett pasientbilde spesifisert av bølgelengde på lampen og visningstallet på kameraet og legge det inn i databasen når det er ferdig kalibrert. Dette gjør at man i dagens kalibreringsprosess kan forholde seg til bare et objekt for å hente ut og sette inn data.

7 Implementasjon

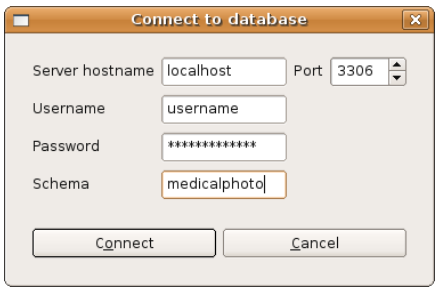
Dette kapittelet inneholder mer informasjon om implementasjon i systemet og kommunikasjon mellom klassene og databasen.

Vil begynne med å se på opplastingsprogrammet og vise hvordan oppdragsgiver vil kunne søke gjennom og vise resultater. Videre vil jeg også se på databaseklassene og klassene til kalibreringsprosessen.

7.1 Opplastingsprogrammet

Opplastingsprogrammet er beskrevet i 6.2.5. Det er to deler av programmet (se Figur 7.6). En automatisert del der man laster opp pasient- og bildedata spesifisert i en metadatafil. Siden denne delen av opplastingsrutinen blir styrt av programkoden kreves det bare å velge en gyldig metadatafil. Dette valget ligger tilgjengelig under fil (File). Den delen av programmet der input til databasen skrives inn manuelt setter større krav til brukeren. De delene av programmet som regnes som manuelle er dem som trenger grafisk grensesnitt spesifisert i første avsnitt av 6.2.5. I samsvar med ønske fra oppdragsgiver skal disse plasseres i en egen avansert (Advanced) meny.

7.1.1 Sikkerhet i programmet



Figur 7.1 Vindu for å koble seg til MySQL databasen

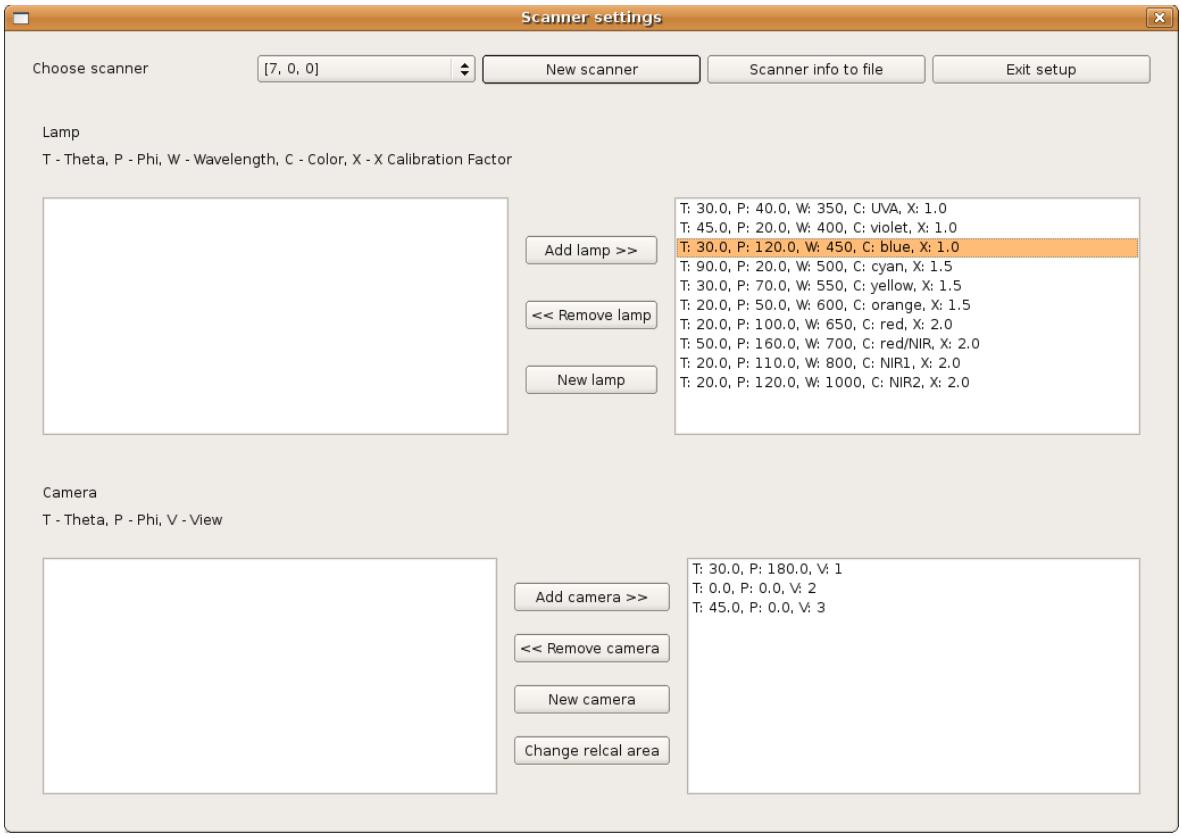
Sikkerhet og brukertilgang var en av grunnene til at man ville oppgradere dagens løsning fra filstruktur. Som nemt over ble den brukerstyrte delen av opplastingen gruppert under avansert i filmenyen for at man skulle unngå at brukerne trykker seg inn ved en feiltagelse.

For at brukerne skal kunne laste opp data med både den automatiske eller brukerstyrte delen av programmet må de logge seg på databasen med både brukernavn og passord som vist på Figur 7.1.

For en MySQL database kan man opprette brukere som får ulike rettigheter til å sette inn, oppdatere og slette data [14]. Om administrator for databasen benytter seg av disse tilgangsrettighetene, vil sjansen minimaliseres for at brukere legger til data som ikke er regnet til deres ansvarsområde.

Dette gir ekstra sikkerhet dersom det sørges for at databasebrukerne bare får tilgang til å legge til nye data på de tabellene de har ansvar for. Når en bruker har koblet seg til MySQL databasen, vil programmet beholde koblingen mot databasen til programmet lukkes. Dette er gjort for å sørge for at det ikke vil bli en unødvendig tidskrevende prosess å legge inn nye data.

7.1.2 Grafisk brukergrensesnitt for skannerinnstillinger



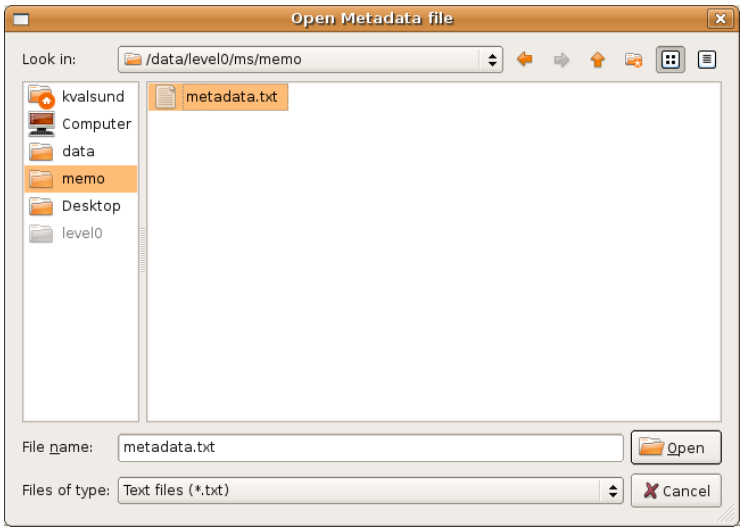
Figur 7.2 Grensesnittet der man kan redigere skannerinnstillinger og legge til nye forekomster av skannere, kamera- og lampeinnstillinger

Figur 7.2 viser det grafiske grensesnittet brukeren skal forholde seg til for å administrere skannerinnstillinger (Scanner settings). Som vist på figuren vil man få opp de valgte skannerinnstillingene i boksene på høyre siden, uvalgte innstillinger er til venstre.

Ved hjelp av en rullegardinmeny kan man velge de ulike skannerne som ligger i databasen. Det er også laget mulighet for å legge til nye skannere, kamera- og lampeinnstillinger.

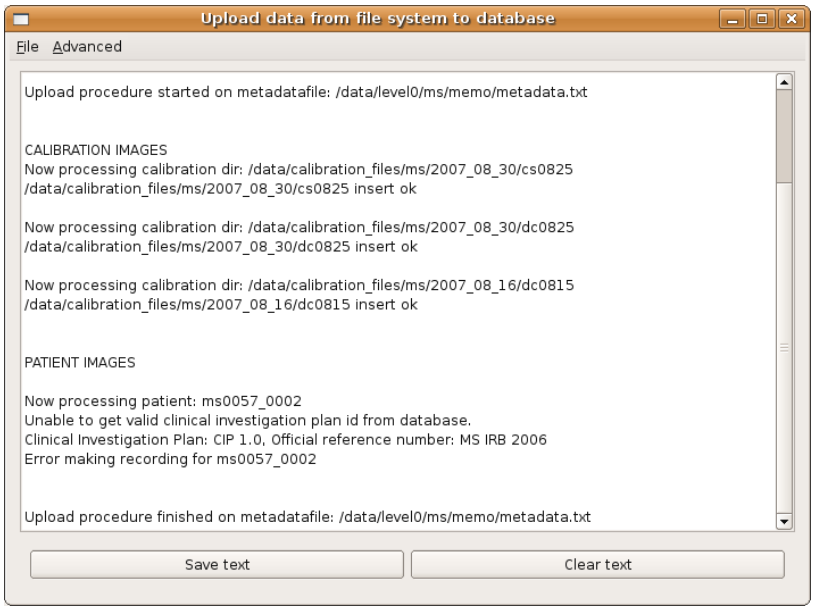
Det er også lagt til en mulighet for å skrive all informasjon om skanneren som er valgt i rullegardinmenyen til fil. Denne egenskapen ble lagt til for at man skulle slippe å måtte koble seg til databasen og/eller bruke opplastingsprogrammet for å vise spesifikasjonene på den gitte skanneren.

7.1.3 Opplasting av data



Figur 7.3 Fildialog for å velge metadatafil

Den automatiserte opplastingsrutinen krever bare at brukeren logger seg på databasen og velger en gyldig metadatafil fra en fildialog som vist på Figur 7.3. Resten av opplastingsrutinen kan observeres i statusvinduet i hovedvinduet. De følgende skjermbildene følger en reell opplastingsprosess, fra det blir spesifisert en metadatafil til dataene er ferdig opplastet.

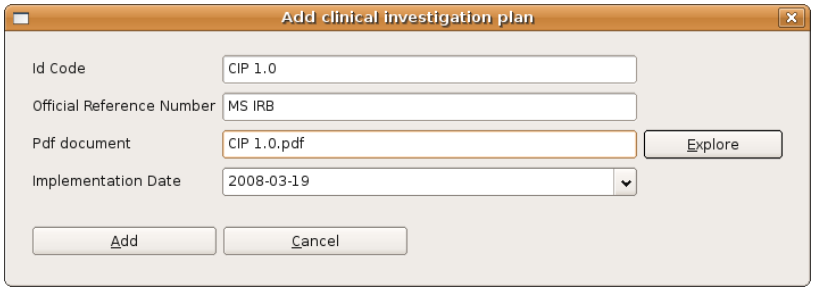


Figur 7.4 Skjermbilde tatt når det mangler klinisk protokoll

Skjermbildet på Figur 7.4 viser at første del av rutinen er å få lastet opp kalibreringsbildene til databasen. Om det blir liggende data om pasienter i databasen som ikke er tilknyttet kalibreringsbilder, vil dette gi problemer for kalibreringsrutinen. Om kalibreringsbildene som er knyttet til en pasient ikke er lagt inn, vil heller ikke pasienten legges inn. I vårt eksempel er

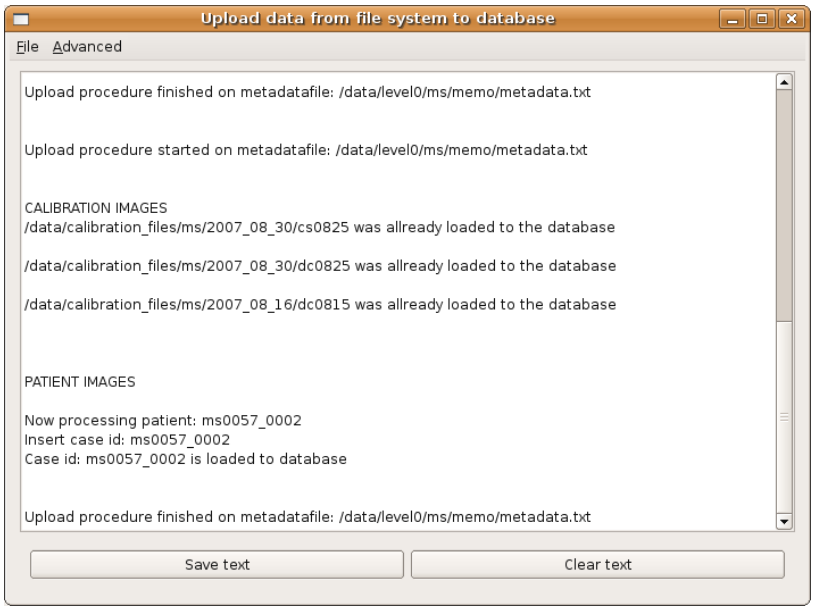
denne delen av rutinen vellykket og det gis tilbakemelding til brukeren at opplastingen er ok. Men når pasientdata skulle lastes var det ikke lastet opp gyldig klinisk protokoll. Resultatet blir at pasienten ikke lastes til databasen, og brukeren får tilbakemelding om mangelen.

Det er lagt til muligheter å lagre tilbakemeldingene man har fått fra den automatiske opplastingsrutinen (Save text) som tekstdokument. Denne egenskapen er nyttig dersom man ikke forstår feilmeldingen og vil videreformidle den til databaseansvarlig eller ikke har tid til å rette den før på et senere tidspunkt. Det kan også være nyttig i vårt tilfelle om bruker ikke har tilgang til den aktuelle kliniske protokollen. Dersom opplastingsrutinen skal kjøres for mange metadatafiler er det mulighet til å fjerne teksten i tilbakemeldingsruten. Dette gjøres ikke automatisk, for å minimere muligheten for at teksten forsvinner uten at brukeren ønsker det.



Figur 7.5 Dialog for å sette inn klinisk protokoll

For å rette feilen man fikk gitt i dette eksempelet må man laste opp ny klinisk protokoll. Dette er en brukerstyrt del av programmet som vist i Figur 7.5. Noen av feltene har begrensinger på hva som kan skrives inn, i dette tilfelle må den kliniske protokollen være ett PDF dokument. Input fra brukeren er en gyldig filsti som viser til et PDF dokument. Datoen med år, måned, dag må skrives i henhold til ISO 8601 standarden. Datoen kan lett velges fra en kalender.

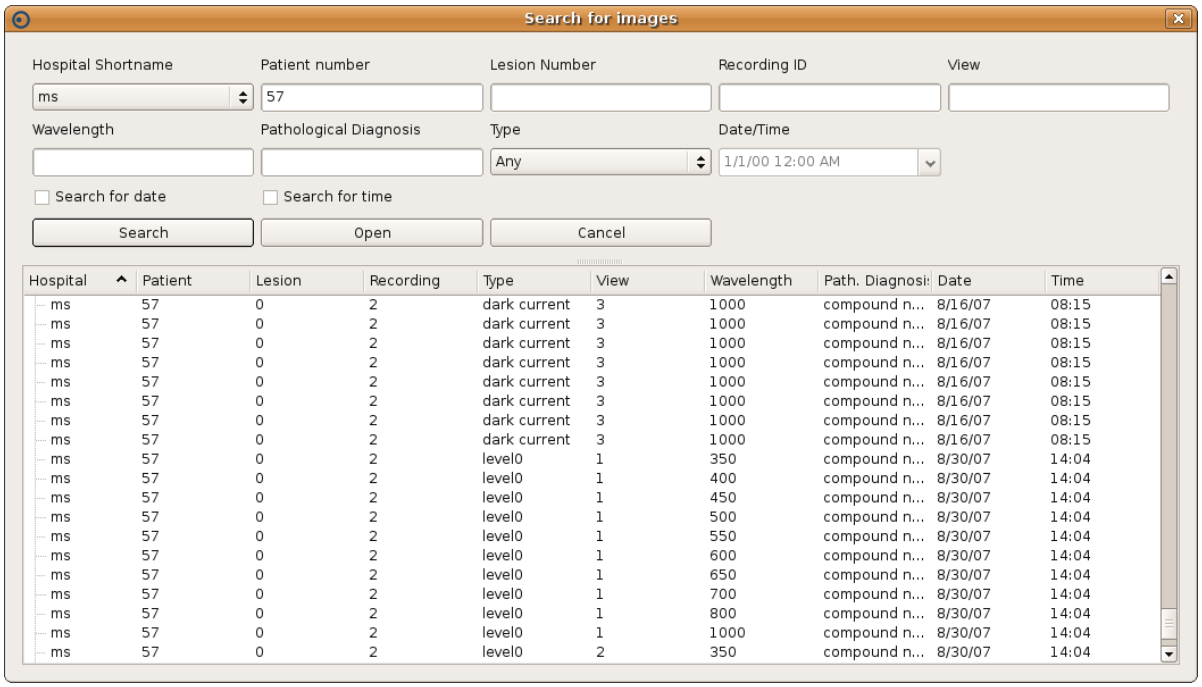


Figur 7.6 Ny kjøring av opplastingsrutine

Etter å ha lagt til klinisk protokoll kjører man den automatiserte prosessen igjen for samme metadatafil. Som vist på Figur 7.6 vil det gis tilbakemelding på at kalibreringsbildene som er knyttet til den aktuelle pasienten er allerede lastet til databasen, deretter lastes også pasient, med all pasientdata og bilder til databasen. Brukeren får tilbakemelding om at opplastingen er vellykket.

7.1.4 Visualisering av bildedata i databasen

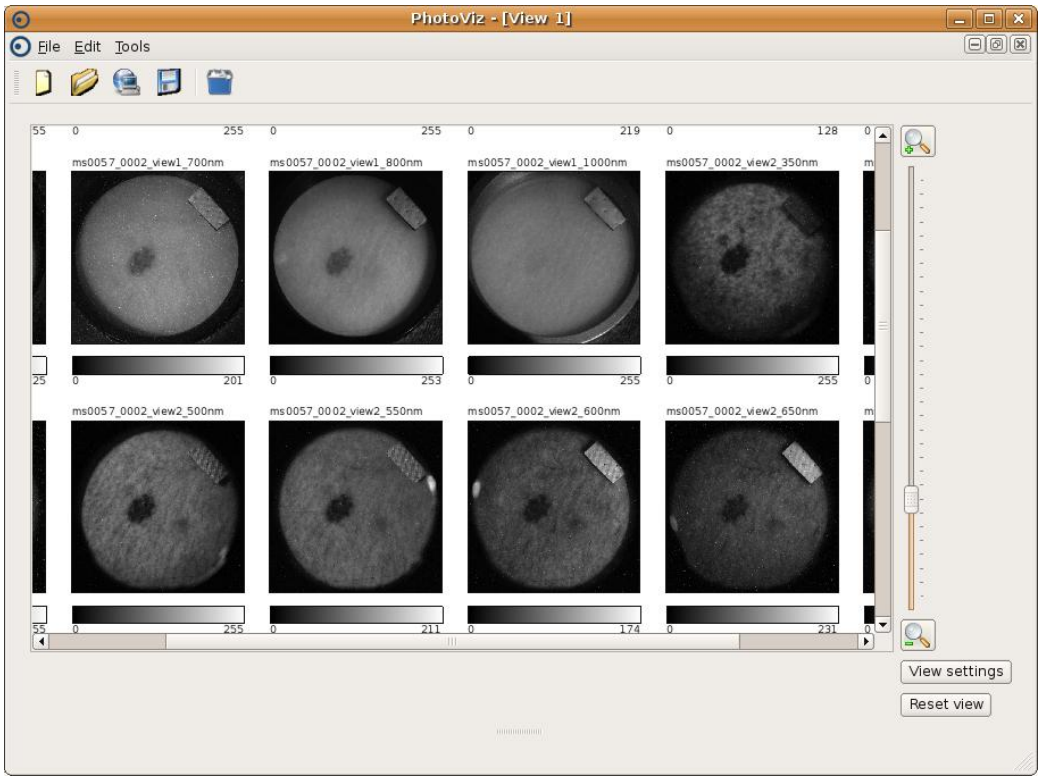
Siden det er vanskelig å vise dataene som er lagret i databasen direkte, må man ha hjelp av et spørresverktøy. Kommandolinje eller MySQL Query Browser kunne blitt brukt, men jeg har valgt å vise dem ved hjelp av visualiseringsprogrammet Photoviz som er blitt utviklet av en annen student parallelt med denne oppgaven [15].



Figur 7.7 Søkefunksjonen i Photoviz [15]

Figur 7.7 viser søkefunksjonen i Photoviz. Ved hjelp av en forholdsvis lett spørring kan man enkelt søke etter bilder spesifisert av en rekke parametere. Etter å søket, vil det komme en linje for hvert bilde som passer til søkeparameterne. Bildene som skal visualiseres merkes i listen, og trykker åpne for å vise dem i visualiseringsprogrammet.

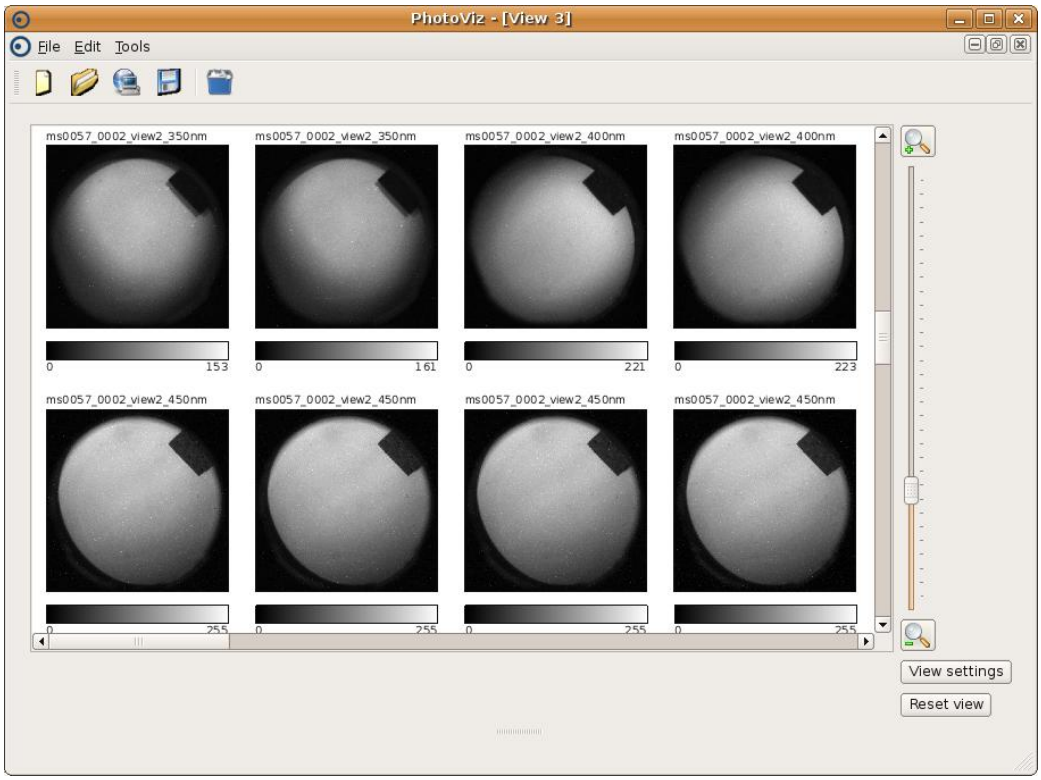
De kommende skjermbildene vil vise ulike typer data som nå lagres i databasen.



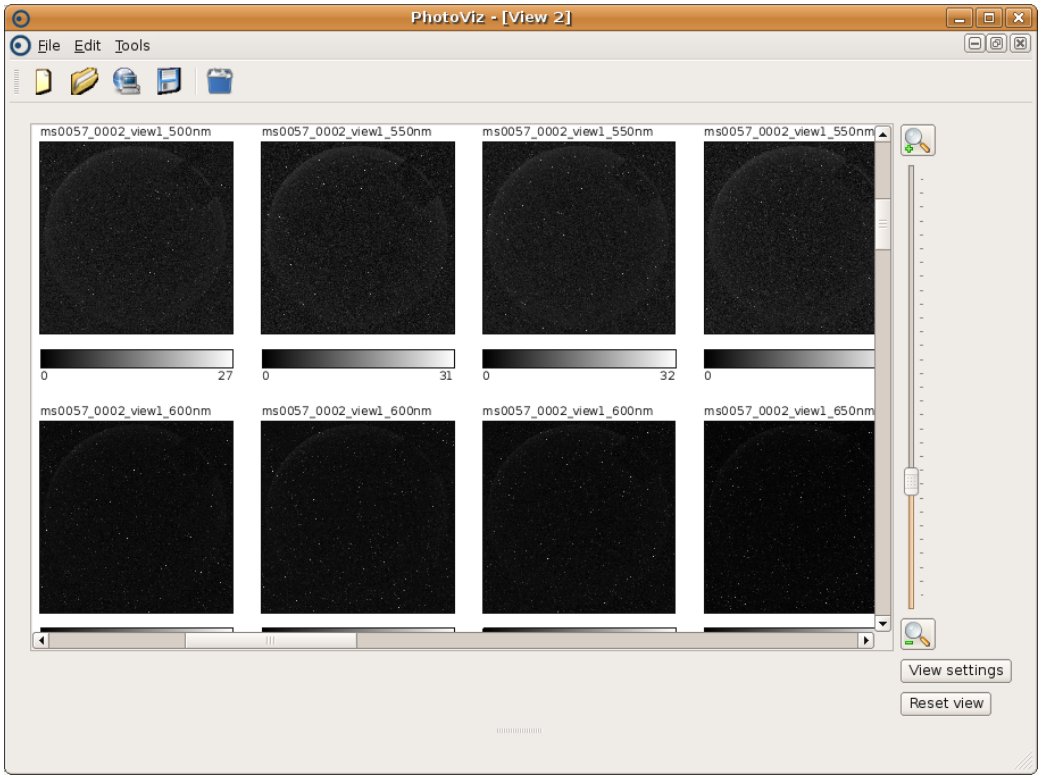
Figur 7.8 Level 0 bilder visualisert ved hjelp av Photoviz [15]

Figur 7.8 viser utvalgte bilder tatt av en lesjon i huden hos en pasient. Firkanten oppe i høyre hjørnet på hvert bilde er tappen som er beskrevet i 6.2.1. Tappens plassering beskrives ved hjelp av det relative kalibreringsområdet. Parameterne som er lagret tilknyttet forkortelse på sykehus, pasientnummer og målingsnummer som er knyttet til bildene blir satt sammen for å gi brukeren den samme opplevelsen av saksidentifikasjonsnummeret som ble brukt i filstruktur. Det samme gjelder kamera plassering og bølgelengde.

Figur 7.9 under viser kalibreringsstandard (CS) bildene som skal brukes for å kalibrere pasientbildene. Også her kan man se tappen oppe i høyre hjørne. Figur 7.10 under viser mørkestrømsbildene (DC) som er knyttet til samme pasient. Disse bildene skulle ha vært helt svarte. Støy vises som lysere piksler.



Figur 7.9 CS bilder visualisert med Photoviz [15]



Figur 7.10 DC bilder visualisert med Photoviz [15]

7.2 Tabeller i databasen og klassene som kommuniserer med dem

Databaseklassene har tilsvarende navn som tabellene i databasen. Noen av dem har blitt blitt noe forkortet for at annen kode som bruker klassene ikke skal bli vanskelig å lese. Det samme gjelder attributter og variabler. Alle attributter som finnes i en tabell, finnes også som variabler i klassen.

Alle tabeller i databasen er implementert med automatisk inkrementerende primærnøkkel. Dette er gjort for at man skal slippe å gjøre spørring på tabellen for å kunne bestemme neste nummer på primærnøkkelen i innsettingsprosessen.

Klassene som er laget for å kommunisere med databasen, utnytter denne egenskapen under innsetting. Metoden med ansvar for innsetting returnerer primærnøkkelen for oppføringen som ble satt inn. Om det skjer noe galt med innsetting vil metoden returnere en verdi som er mindre enn 1. Feil kan skyldes at det ikke finnes gyldige oppføringer i de tabellene det refereres til i fremmednøklene. Andre mulige feil er at man har ugyldige datatyper, eller at det er problemer med tilkoblingen til databasen.

Opplastingsprogrammet benytter seg av de returnerte primærnøklene. Et eksempel er når det skal legges til bilder i en samling. Første steg er å lage samlingen. Når samlingen settes inn i databasen returneres primærnøkkelen den har fått. Bildene som skal tilknyttes samlingene opprettes med samlingens primærnøkkel som parameter. Når bildet legges i databasen vil det ha samlingens primærnøkkel som fremmednøkkel, som sørger for at koblingen mellom dem er opprettholdt. En lignende metode brukes når man setter inn data for en pasient.

Klassene som kommuniserer med databasen er primært laget for innsetting, men vil for en del tabeller være aktuell som uthentingsklasser. Utfordringen med hensyn på uthenting av data fra databasen er at databaseklassen bare har ansvar for en tabell i databasen. En typisk spørring vil involvere flere tabeller. Allerede i Case id klassen som blir beskrevet nærmere i neste delkapittel ble det aktuelt å lage mer sofistikerte spørringer enn det databaseklassene kan tilby. Fordelen med databaseklassene er at om man bare spør etter primærnøkkel i en mer komplisert spørring, kan de brukes til å hente resten av informasjonen om oppføringen i tabellen. Denne egenskapen kan vise seg å være nyttig spesielt knyttet mot spørringer som er rettet mot bildetabellen. Den kunne ha blitt benyttet av søkefunksjonen i Photoviz (se Figur 7.7). Der er det først en spørring som henter ut informasjon som kan identifisere bildet og vil senere hente ut utvalgte bilder.

7.3 Klasser til kalibreringsprosessen

Den gamle kalibreringsrutinen ble utført i flere MATLAB – skript. Parallelt med denne oppgaven har en medstudent [16] utviklet en kalibreringsrutine i C++ som skal erstatte MATLAB - kalibreringsskriptet. Denne kalibreringsrutinen skal kunne benytte seg av databasen for å hente ut data.

Case Id klassen som omtalt i 6.2.6, skal stå for databasekommunikasjon for kalibreringsprosessen. Case Id er den engelske oversettelsen av saksidentifikasjonsnummer som er blitt nevnt tidligere. Ansvarsområdene for klassen er først og fremst å hente ut og sette inn bildedata knyttet til ett saksidentifikasjonsnummer, men det ble også implementert funksjoner som skal kunne brukes for kalibreringsdata (CS, DC og kalibreringsmatrise).

Den viktigste delen av Case id klassen er å kunne hente ut ukalibrerte pasientdata, og sette inn kalibrerte pasientdata.

Som en del av prosessen for å kalibrere pasientdataene vil kalibreringsrutinen produsere kalibreringsmatriser ut fra CS og DC bildene, som forklart i 6.2.6. Kalibreringsmatrisene blir først og fremst laget for å fjerne overeksponerte piksler, bakgrunnspiksler og støy på pasientbildene. Genereringen av kalibreringsmatrisene er tidskrevende og vil derfor lagres for å unngå at kalibreringsmatrisen måtte kalkuleres på nytt for hver pasient som har samme kalibreringsbilder.

Case id klassen fikk derfor mulighet til å legge inn kalibreringsmatriser i en kalibreringsmatrisesamling. Disse gjøres søkbar etter de samme parameterne som kan brukes for å finne CS- eller DC – samlingene som ble brukt som input.

8 Evaluering

Dette kapittelet skal evaluere de ulike sidene av prosjektet. Begynner med databasen siden løsningen her ligger til grunn for de andre delproblemene, deretter vil jeg se på opplastingsprogrammet før jeg avslutter med objektene som ble brukt av kalibreringsprosessen.

8.1 Databasen

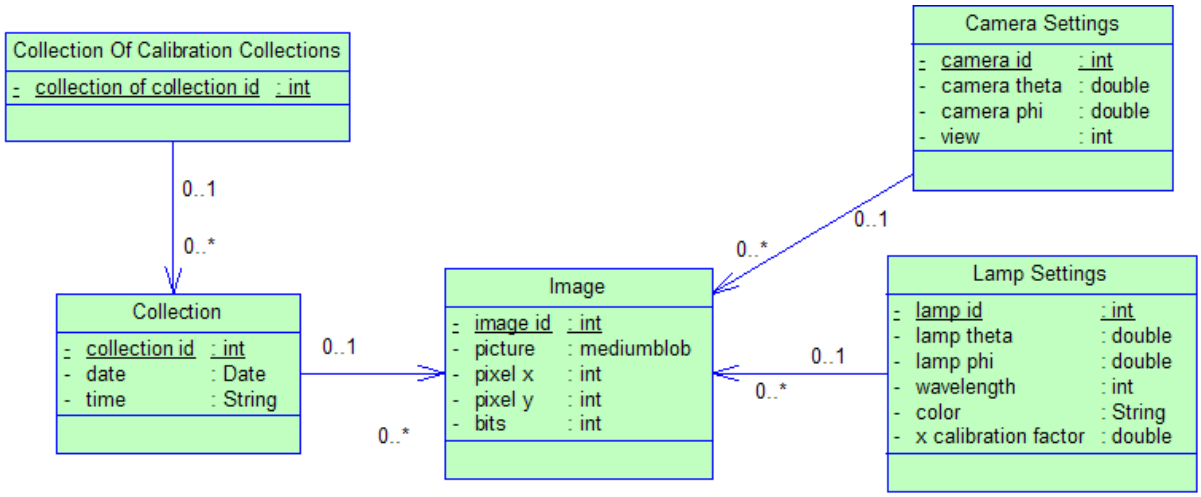
Det viktigste når man skal endre lagringssystemet fra filstruktur til database er å få bevart alle verdier, det vil si attributter og koblinger som ligger i nåværende system er ivaretatt og lagret på korrekt format i den nye databasen. Det neste man må utføre er å få strukturert attributtene i tabeller slik man unngår dobbellagringen av data som i høyeste grad forekommer i det nåværende systemet i filstrukturen. Det er ikke til å legge skjul på at uansett hvordan man effektiviserer lagringen av dataene, er det bildene som lagres som vil ta mest plass. De andre dataene vil bare utgjøre en brøkdel av det som lagres i bildedata. Ett CS -, DC- eller level 0 bilde tar omtrent 1,2 MB, et level 1 bilde tar omtrent 2,3 MB og en kalibreringsmatrise tar omtrent 7 MB for hvert bilde. Med andre ord, de andre feltene vil kunne lagres mange ganger før man vil få prosentvis økning på hva som lagres i databasen. Dette betyr ikke at man skal glemme all normalisering, men det kan være greit å ha det i tankene når man designer systemet.

8.1.1 Denormalisering: Optimalisering vs. normalisering

Om man skal følge normaliseringsprinsippene slavisk, bør man trekke ut alle attributter som er felles for flere forekomster i en tabell. Når man følger disse prinsippene kan man i utgangspunktet begynne med en stor tabell som inneholder alle attributter og skille ut dem som hører sammen. I praksis vil man ikke gjøre det slik, men prinsippene som ligger bak er verdt å rette seg etter om man vil ha god design på databasen. Det finnes også unntak når man skal normalisere, der man ender opp med mer kompleks design, som er mindre intuitiv og vanskeligere å forholde seg til om man skal følge prinsippene til punkt og prikke. I mitt system finnes det noe slike, og jeg vil nå trekke dem frem og forklare hvor de ville bli plassert dersom jeg fulgte normaliseringsprinsippene, hvor jeg plasserte dem og hvorfor de ble passert der de ble.

Det første jeg vil dra frem er kalibreringsmålinger. Som det blir beskrevet i kapittel 3.1 blir det for kalibreringsmålinger tatt 10 bilder med hvert kamera for hver bølgelengde. De blir gitt et sekvensnummer fra 0 – 9. Siden man da logisk for hvert sekvensnummer får 30 bilder, ett bilde med hvert kamera for hver bølgelengde, kunne man om man følger normaliseringsreglene [17] tatt ut attributtet som beskriver sekvensnummeret (calibration measurement) til samlingstabellen (Collection). Problemet ville da vært at sekvensnummer ikke har noen betydning for verken pasientsamlingene (level 0 og level 1), kalibreringsmatrisesamlingene. Det er rimelig å anta at sekvensnummeret ikke vil ha noen logisk betydning for verken fysiologidata eller morfologidata når de skal bli en del av

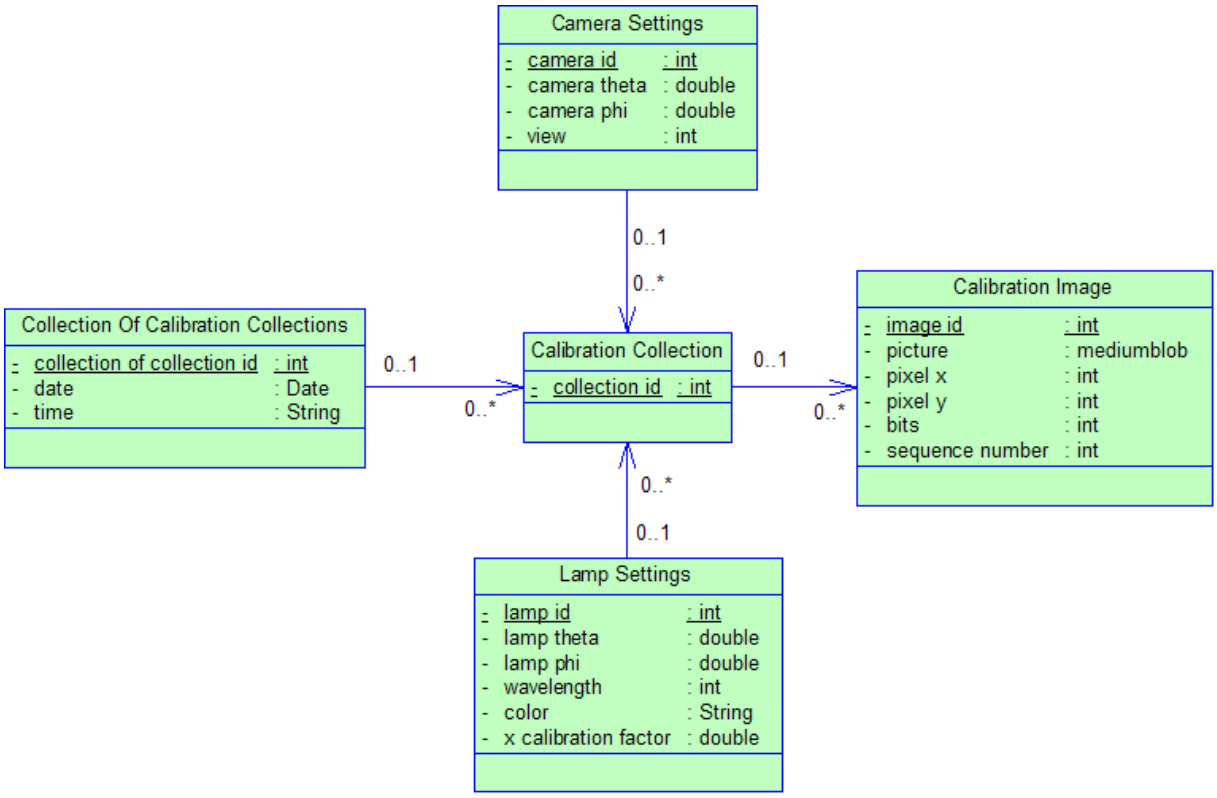
databasen. En annen ting å tenke på er at man da ville logisk delt opp noe som var sett på som en samling i filsystemet, til å være 10 samlinger i databasen. En annen negativ effekt er at det ville blitt en mer omfattende operasjon å knytte pasientmålingene mot rette kalibreringssamlinger. For hver pasient vil det bli 10 ganger så mange koblinger for hvert kalibreringssett de knyttes mot. Kalibreringsmålinger vil på sikt bli tatt en gang hvert kvartal, halvår eller år. Kostnaden av å lagre koblingen mellom måling av en lesjon (Recording) og kalibreringssamlingene vil bli høyere enn det man sparer inn på å fjerne dobbellagring av sekvensnummeret i kalibreringbildetabellen.



Figur 8.1 Alternativ løsning med samling av samlinger tabell for kalibreringssett

For å løse problemet med antall koblinger mellom målingstabellen og samlingene av kalibreringsbilder kunne man laget en ny tabell som er en samling av kalibreringssamlinger som vist på Figur 8.1. Denne løsningen ville samlet kalibreringsdataene til en enhet som ligner den man har i filstruktur. Denne ”samling av kalibreringssamlinger” - tabellen ville vært lite nyttig for pasient- og kalibreringsmatrisesamlingene. Man kunne enten trikset det til med å koble alle samlinger uansett type til denne nye ”samling av samlingstabeller” - tabellen, eller fått et brudd på hvordan man refererte til ulike typer data. CS- og DC - bilder refereres til på en måte, pasientbilder og kalibreringsbilder på en annen måte.

Et eventuelt brudd på hvordan man refererer til ulike typer data ville gjort spørringene unødvendig komplekse for visualiseringsverktøyet som skal bruke databasen. Søkefunksjonen i dagens visualiseringsverktøy kan man gjøre en enkel spørring for å hente ut alle typer data knyttet til en pasient. Dette måtte ha blitt løst med en mer kompleks spørring om det var ulike måter å lagre samlinger på i databasen. Et annet punkt er at dato og tid burde vært plassert i den nye ”samling av kalibreringssamlinger” tabellen for å følge normaliseringsprinsippene. Dette ville i hvert fall gitt et brudd på hvordan man refererer til samlinger av CS- og DC bilder og samlinger andre bilder.



Figur 8.2 Alternativ løsning der kamera- og lampeinnstillinger knyttes til samlingen

I kalibreringsprogrammet ønsker man å hente bildene med hensyn på kameraposisjon og bølgelengde fremfor sekvensnummer. Sekvensnummerets betydning i dagens database er begrenset. Man vil bare kunne luke ut bilder med sekvensnummer 0 siden de har vist seg å forringe kvaliteten på kalibreringsprosessen.

Spørsmålet er da om det da hadde vært lurere å dele inn kalibreringssamlingene med hensyn på kameraposisjon og bølgelengder (se Figur 8.2), dersom man først skal dele opp det som logisk sett var en samling av kalibreringsbilder tidligere. Da ville CS- og DC – bildene vært koblet logisk, i hvert fall for kalibreringsprosessen. For å følge normaliseringsprinsippene bør koblingen mot kamera- og lampeinnstillingene ivaretas i den nye kalibreringssamlingstabellen. Dette ville ført til at man opererte med 30 oppføringer den nye samlingstabellen, der man tidligere bare hadde 1 samling i filstrukturen. Lagringsstrukturen vil skille seg fra den som er brukt level 0-, level 1- og kalibreringsmatrise – samlingene. Man måtte ha laget en ny bildetabell bare til bruk for CS- og DC bilder, siden koblingen mot kamera- og lampeinnstillingene for disse blir ivaretatt i kalibreringssamlingstabellen istedenfor å bli ivaretatt i bildetabellen. For visualiseringsprogrammet vil dette gi de samme konsekvenser som ved den andre alternative løsningen.

Konklusjonen er at for å unngå unødvendig kompleksitet i databasen løses problemet med sekvensnummer ved hjelp av arv på bildetabellen. Selv om resultatet er at sekvensnummeret lagres flere ganger enn nødvendig, gjør den strukturen på databasen mye mer oversiktlig, og databasen blir enklere å bruke.

En annen ting som kunne trekkes ut som egen tabell, er bredde (pixel x) og lengde (pixel y) på bildet som lagres i bildetabellen. Disse dataene er i høyeste grad standardiserte og vil være en verdi som er like for alle bilder tatt med en skanner for en gitt skannervektor. Lengde og bredde på bildene endres bare når en foretar en ny geometrisk kalibrering. Skulle en fulgt normaliseringsreglene burde disse verdiene bli lagret i skannertabellen. Et annet moment med lengde og bredde på bildet er at det lagres i header i bildefilen. Det vil si at de ligger egentlig allerede i databasen, men er utilgjengelig med mindre man henter ut bildet. Grunnen til at de blir plassert i bildetabellen er langt mindre komplisert enn utledningen om kalibreringssamlingene. Brukerne av databasen vil lage sine egne spørringer i for eksempel MySQL Query Browser - programmet. Det er høyst tenkelig at de vil lage spørringer der de vil hente ut informasjon om bildet, men er lite interessert i hvilken skanner som er brukt. Tilbakemeldingene jeg har fått fra oppdragsgiver er at det er enklere om man plasserer lengde og bredde til bildetabellen, slik at man ikke skulle trenge å koble mot skannetabellen for å få informasjon om størrelsen på bildet.

Lagring av lengde og bredde i bildetabellen gjør at man lagrer samme felt mange flere ganger enn nødvendig. Men for brukerne oppleves situasjonen som enklere og jeg har derfor implementert dem nettopp i bildetabellen. Dette kan også bli nyttig dersom man skal spore eventuelle avvik i størrelsen på enkelte bilder i noen samlinger. Det er ikke et tema i dagens system, men kan kanskje bli det ved fremtidige endringer i programvaren som er knyttet til skanneren. Opplastingsprogrammet henter størrelsen på bildet rett fra headeren, så man er sikret at lengde og bredde som ligger i databasen er tilknyttet de faktiske størrelsene på bildet.

Et annet felt i bildetabellen som med hensyn på normaliseringsprosessen er attributtet bits. Dette attributtet kunne vært lagret under samlingstype (Collection Type). Den ville kanskje bli regnet som irrelevant for fysiologi og morfologidata, men er relevant for nesten alle typene som ligger inne i dag. Unntaket er for kalibreringsmatrisene. Logikken i å plassere dem i bildeklassen er tilgjengelighet for spørringer, som for lengde og bredde.

En annen standardisering som kan bli aktuell i fremtiden er nevnt tidligere i delkapittel 6.2.3, er å lage en tabell for leger. Nåværende programvare på klinikkene lar legene skrive inn fritt det de vil under feltet leger. I forskningsøyemed kan det være greit å ha oversikt over dem som var med på prøven. Men om en begynner å standardisere input i programvaren til skanneren vil det være naturlig å lage en egen legetabell.

8.1.2 Andre kommentarer om databasen

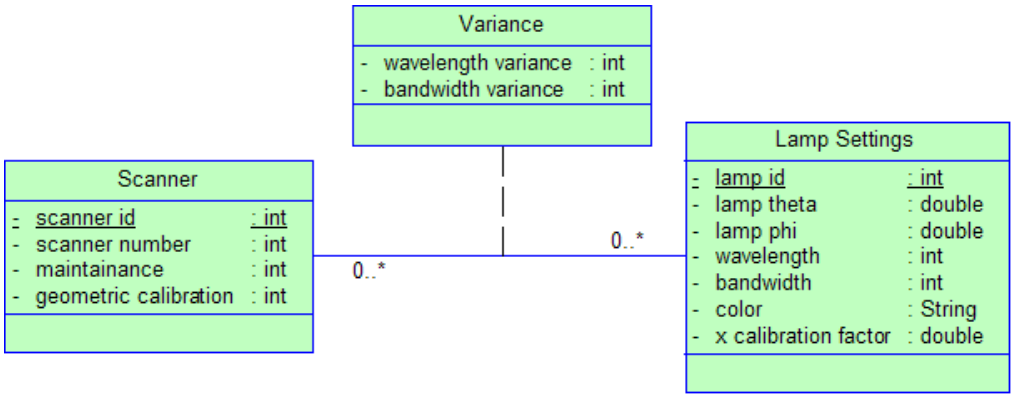
I 6.2.3 drøftet jeg også hvorfor jeg plasserte koblingen til klinisk protokoll til lesjon og ikke pasient. Her er begrunnelsen at man skal være sikker på at dataene man har lagret i databasen er å stole på, og at man ikke får lagret feil klinisk protokoll knyttet til måling av en lesjon.

Samlingsrolle (Collection Role) er også en tabell man kan merke seg. Denne knyttes til mange til mange koblingen mellom lesjonsmålingstabellen (Recording) og samlingstabellen (Collection). Har tidligere i oppgaven fortalt om DC2 samlinger som egentlig er en DC samling som opptar en annen rolle for enkelte lesjonsmålinger. For å slippe å lagre hele samlingen på nytt der samlingstypen (Collection Type) er ”dark curent 2” istedenfor ”dark current” knyttet jeg enkelt og greit en tabell til mange til mange koblingen mellom måling og samling i databasen. Attributtet som ble lagt til, ble kalt rolle (role).

I mange til mange koblingen vil rolleattributtet settes til 1 for alle DC-samlinger som har rollen DC2 for den aktuelle lesjonsmålingen. For alle andre koblinger settes attributtet til 0.

Attributtet rolle kunne for dagens system vært en boolsk verdi, men valgte å lagre det som en heltallsverdi for å ta høyde for lignende situasjoner på andre samlingstyper.

Siste punktene som er aktuelle ble jeg gjort oppmerksom på av oppdragsgiver så tett opptil innlevering av oppgaven at jeg fikk ikke tid til å rette på det. Dette er punkt som ville blitt brukt og implementert i databasen om de hadde blitt kjent tidligere. For lampene lagres en bølgelengde som er fast. Den faktiske sitasjonen er at bølgelengden kan variere. Etter systemet var ferdig implementert med ferdig opplastingsprogram og database ble det informert om at det vil være en viss variasjon i den faktiske bølgelengden på LED lampene som kjøpes inn. Dette er tilfelle selv om spesifikasjonen fra produsent på lampenes bølgelengde er lik fra gang til gang. Det vil si at selv om man kunne ønske at alle bølgelengdene skulle være like, vil de i praksis ikke være det. Denne variasjonen i bølgelengdene kan måles og avviket burde vært lagret i databasen. Hadde man implementert databasen som vist på Figur 8.3, der man knytter en varianstabell til mange til mange koblingen mellom skanner og lampeinnstillinger, ville man kunne ha nøyd seg med de 10 lampeinnstillingene og registrert avvik. Det som blir tilfellet i dagens databasestruktur er at man må legge til en ny oppføring i lampeinnstillingstabellen der den faktiske bølgelengden blir lagret.



Figur 8.3 Ønsket kobling mellom skanner og lampeinnstillinger

Det man også kan merke seg på figuren er at det er blitt introdusert en helt ny variabel, båndbredde (bandwidth). Dette er en variabel som ikke blir lagret i nåværende filstruktur, men man har funnet ut at det ville være nyttig å lagre. Muligheten for båndbredde og variasjon i båndbredde ble jeg først introdusert for på samme tid som jeg ble gjort oppmerksom på variasjon i bølgelengden.

Båndbredden: Oppdragsgiver ønsker å få lagret båndbredden på FWHM (Full Width Half Maximum). Kort fortalt er dette hvor stor forskjell det er i bølgelengden på lyset som blir sendt ut av en lampe når den lyser ved halv intensitet. Dette er en egenskap som beskriver hvor god fargepresisjonen er i lyset som blir sendt ut av lampen.

8.1.3 Eventuelle endringer i databasen

Et alternativ til å lagre bildene i databasen var å lagre filstien til bildene på serveren. Databasen kunne da bli brukt først og fremst for å lagre all annen informasjon enn selve bildene, og relasjonene mellom data. Opplastingsrutinen ville da kunne lastet opp data beskrevet i metadatafilen på kort tid, og man ville fremdeles kunne gå gjennom bildene i filstruktur om det var ønskelig.

Ulempene med å lagre bare filsti til bildene er at man da fremdeles ville vært sårbare for brukerendringer på filstystemet. Om man regulerte dette ved hjelp av tilgangsrettigheter har man da to systemer å holde orden på. Dataene fra de ulike sykehusene kommer dessuten på ekstern harddisk. Tiden det tar å laste opp data til databasen fra ekstern harddisk er omtrent den samme som det tar å laste over til filstruktur. Så da ville den samlede tiden det tar å kjøre opplastingsrutinen uansett være den samme, om ikke noe lenger.

Andre problemer med denne løsningen er for programmene som skal benytte seg av databasen. Visualiseringsprogrammet og kalibreringsprogrammet hadde da blitt begrenset til å kjøre på selve serveren. Om de ikke da måtte ha implementert program eller nye metoder for å overføre filer fra serveren spesifisert av filstiene i databasen. I verste fall ville brukerne måtte logge seg på to systemer for å få tilgang til de data de ønsket. For å sørge for at andre program ikke blir låst til å måtte kjøres på serveren, samt sikkerhetshensyn gjorde at databasen lagrer bildefilene og ikke filsti.

8.2 Opplastingsprogrammet

8.2.1 Generelt om opplastingsprogrammet

Opplastingsprogrammet er i dag et program primært for å laste opp data. Eneste brukeren har mulighet for å endre på er kamera- og lampeinnstillingene for en skanner. Innunder å kunne endre på kamerainnstillinger menes det også at man skal kunne justere koordinatene til det relative kalibreringsområdet. Grunnen til at det har vært hovedfokus på opplasting av data er naturlig nok å få inn dataene fra filstrukturen til databasen. Siden det er to andre studenter som har vært avhengig av en ferdig database med reelle data for at de kan implementere den delen av programmet deres som har databasetilknytning har denne biten vært viktig. Kravspesifikasjonen spesifiserte bare et opplastingsprogram.

En annen grunn til at det har vært lite fokus på mulighet for å endre data var en enighet tidlig om at de som skulle ta seg av opplastingsrutinen, samt legge til kliniske protokoller, eventuelle kliniske diagnoser, samlingstyper eller skannere, derunder kamera- og lampeinnstillinger skulle være brukere med kjennskap til systemet, og at man kunne forutsette at vedkommende som har ansvar for å sette inn nye data gjør dette nøyaktig. Disse oppgavene som ikke er automatiserte vil typisk være rutiner som gjøres svært sjeldent. Den operasjonen som gjøres oftest er nettopp å legge til nye skannervektorer og legge til innstillinger til dem. Siden det i nåværende databasesystem ville kunne lagres bølgelengder som er ganske like vil man også kunne fjerne innstillinger til en skanner for å ta høyde for at brukeren kan gjøre feil når det blir lagt inn nye innstillinger.

Selv om det på sikt bør lages muligheter for å kunne editere data også for andre klasser enn skanner, vil jeg komme med en liten kommentar om hvordan det kan løses om man gjør noen feil. Siden ingen av de grafiske innsettingsmetodene kommuniserer med mer enn en tabell, og tabellene det er snakk om vil ha et begrenset antall oppføringer, vil man som en midlertidig metode kunne bruke MySQL Query Browser. Den nyeste versjonen har lagt til muligheten for å editere data. Siden man opererer med bare en tabell vil man kunne skrive så enkelt som SELECT * FROM ”aktuell tabell” og redigere feltet/feltene der man har skrevet inn feil verdier.

En viktig del å tenke på når man skal editere i databasen er at man bør man ha god kontroll på hvilke felter som blir editert, om ikke kan man i verste fall ende opp med at koblingene i databasen blir feil, eller at parametere som brukes av program forsvinner. Oppgaven med å gi mulighet til å endre på data kan alt ettersom hvilke felt man ønsker å endre, bli ganske omfattende.

8.2.2 Klassene som kommuniserer med databasen

Det er mange klasser som har ansvar for kommunikasjon med databasen. De har mange like egenskaper og på dette grunnlag falt valget på å evaluere dem samlet.

For hver tabell i databasen finnes det en klasse med samme navn som har ansvar for innsetting av data til databasen å hente ut data fra databasen ut fra gitte søkeparametere. Man kan være sikker på at alle attributtene i tabellene er medlemsvariabler i klassen med samme navn. Tanken bak å ha strukturen slik er at nye mer sammensatte klasser skal kunne benytte seg av databaseklassene for kommunikasjon med databasen. En annen fordel er at om man ønsker å lage et nytt program som skal kunne kommunisere med databasen, kan man lett identifisere hvilke klasser en trenger ut fra å se på modellen av databasen. Dette ble aktuelt for klassen som brukes i kommunikasjon med databasen av kalibreringsrutinen som nevnes i 8.3, samt for selve opplastingsprogrammet.

Alternativet til å ha mange små klasser kunne vært å ha en stor klasse som håndterte all databasekommunikasjon. Allerede med de programmene som benytter seg av databaseklassene i dag, viser at dette kunne blitt unødvendig tungt. For klassen til kalibreringsprogrammet som bare trenger et utdrag av databaseklassene, ville man hatt mye unødvendig funksjonalitet. I opplastingsprogrammet gjør man relativt små innsettinger underveis som dataene er blitt validerte. En annen faktor er utvidelser. Å lokalisere hvor en skal gjøre endringer i en stor tung databaseklasse er mer vrient enn å endre for en klasse med tilsvarende variabler som det er attributter i databasen.

Men det er en databaseklasse som har fått mer funksjonalitet enn bare sette inn og hente ut oppføringer i tabellen med tilsvarende navn, nemlig skannerklassen. Skannerklassen er en sentral klasse både i opplastingsprogrammet og i kalibreringsrutinen. I opplastingsprogrammet brukes den både for å administrere koblingene mot kamera- og lampeinnstillingene. Dette gjør at klassen kan bli sårbar for endringer ikke bare i skannertabellen, men også i relativt kalibreringsområde- kamerainnstillings- og lampeinnstillingstabellene. Grunnen til at jeg legger til denne funksjonaliteten i skannerklassen, er for å kunne tilby samme funksjonalitet i den som i MATLAB – skriptet GetPtInfo (beskrevet i 6.1.1).

8.2.3 Bruk av grafisk grensesnitt for innsetting av data i databasen

Som nevnt over er det en klasse som har ansvar for å sette inn nye forekomster av en tabell i databasen. En modell av hvordan hvilke klasser som kommuniserer med hverandre er vist i Figur 6.6. Dette gjør at innsetting av en oppføring til en tabell er lett. Løsningen var ganske enkelt å lage et vindu der brukeren får mulighet til å skrive inn de data som påkreves i databasen. De fleste av dem er meget enkle. Nytten av en mer kompleks skannerklasse viser seg når man skal legge til skannerinnstillinger. Å legge til en ny skanner er bare å legge til tre tall for vektoren. Men skannervektoren er ikke nyttig i seg selv. Det er skannervektorens relasjoner til kamera- og lampeinnstillinger som er viktig. Som nevnt over har skannerklassen funksjonalitet til å legge til og fjerne innstillinger, men den har også muligheten for å hente hvilke kamera- og lampeinnstillinger den bruker, og ikke bruker. Disse metodene kan brukes i et grafisk grensesnitt. I det grafiske grensesnittet kan merke de innstillingene man ønsker å

legge til og trykke legg til, eller når man skal fjerne, merke dem man vil fjerne og trykk fjern, og skanner klassen tar seg av koblingene i databasen.

Utviklingen av grensesnittet har blitt evaluert underveis av oppdragsgiver, slik at det skal oppleves intuitivt og lett å bruke.

8.2.4 Klassen som har ansvar for opplasting av data spesifisert i metadatafil

Dagens struktur gir opplastingsklassen (UploadToDatabase) mye ansvar. Unntakene er oppgaven å tolke metadatafilen, som er skilt ut til en egen klasse, og databaseklassene. Dersom det hadde vært mer tid til rådighet ville flere av metodene i opplastingsklassen bli skilt til flere klasser som hadde hatt ansvar for ulike områder av dataene. Metodene jeg tenker på er dem som har ansvar for å validere en samling (Validate Collection), sette inn informasjon om pasient (Insert Patient Info), sette inn pasientbilder (Insert Patient Images) og sette inn kalibreringsbilder (Insert Calibration Images).

Validering av samling (ValidateCollection)

Denne klassen ville hatt ansvar for å validere om filene som ligger i en mappe er tilstrekkelig for å kunne kalles en samling. Innparameterne ville vært mappen som skal undersøkes, der det et allerede er lagt til filter for å sortere ut de filene som er relevant. I dagens system er dette at det er tiff – bilder, og utvalgte deler av filnavnet alt ettersom det er en pasientsamling eller kalibreringssamling som skal vurderes. Klassen skulle talt opp at det er et tilstrekkelig antall filer, sjekket at filene er lesbare og ikke er under en viss størrelse.

Sette inn informasjon om pasient (InsertPatientInfo)

Denne klassen ville hatt ansvar for å legge inn data knyttet til en pasient. Det omfatter klassene pasient, lesjon og måling, men også ansvar for å legge til koblingene til samlingstabellene. En videre utvilking av CaseId (saksidentifkasjonsnummer) klassen kunne nok blitt brukt i denne løsningen.

Sette inn kalibreringsbilder (InsertCalibrationImages)

Dette er en metode i dagens opplastingsklasse. For en bedre struktur burde denne vært skilt ut som en egen klasse. Denne klassen ville benyttet seg av validering av samling klassen for å få godkjent dataene i filstruktur. Denne klassen ville kunne brukes til å laste opp CS og DC samlinger til databasen som det ikke refereres til metadatafilene.

Sette inn pasientbilder (InsertPatientImages)

Denne metoden ville hatt mange av de samme egenskapene som innsetting av kalibreringsbilder. Men da filnavn og hvilke bildetyper som skal settes inn i databasen er ulike ville det vært mest ryddig å skille disse metodene i forskjellige klasser. Et annet viktig moment med denne klassen er at man ikke bør legge inn pasientbilder i databasen før informasjonen om pasient-, lesjons- og målingsnummer finnes i databasen. Også denne klassen ville brukt validering av samling klassen for å validere dataene som finnes i filstruktur.

8.2.5 Signals og slots i Qt

Det kunne vært aktuelt å bruke Observer mønsteret [18] for å sende tilbakemelding til det grafiske grensesnittet om hva som skjer i den automatiske opplastingsprosessen. Men dette ble unødvendig siden Qt har mulighet for å benytte signals og slots. Klassene som står for håndtering av data sender ut signaler som en streng som informerer om hva programmet holder på å laste opp, og en status på opplastingen etter den har lastet opp. Om det har skjedd noe feil vil den informere om hvor feilen ligger. Det grafiske grensesnittet setter en slot til å lytte på signalene som sendes ut av opplastingsklassen. Det som ankommer sloten vises på skjerm.

8.3 Klasser til bruk for kalibreringsprosessen

Som jeg nevnte i 8.2.2 vil man for hver tabell ha en klasse som bruker enkle operasjoner for å sette inn og hente ut data. Man har i kalibreringsprosessen tatt i bruk enkelte av disse. Som tidligere nevnt ble det også laget en klasse kalt Case Id som refererer til saksidentifikasjonsnummeret. Klassene som er knyttet til tabeller i databasen vil ikke bli evaluert her bortsett fra skannerklassen. Fokuset vil være mot Case Id objektet. Evaluering av denne klassen er litt vanskelig, siden bruken av den ikke er knyttet til oppgaven min. Det jeg vil trekke frem er opplevelsen til medstudenten av å bruke databaseklasser, i forhold til å måtte forholde seg til dagens system med filstruktur.

8.3.1 Input av hvilke saksidentifikasjonsnumre som skal kalibreres

Case Id klassen genererer ved hjelp av en statisk metode en liste over alle saksidentifikasjonsnumre som ikke er blitt kalibrert. Tilsvarende når man kjører mot filsystem må denne listen bli gitt som input til programmet. Dette gjør at saksidentifikasjonsnumrene enten må skrives inn en etter en, eller spesifiseres i en tekstfil. Dette fører til at den nye løsningen med Case Id klassen enklere ville kunne automatisere kalibreringsprosessen.

8.3.2 Bruk av skannerklassen

Som nevnt tidligere er det et mål om å frigjøre seg fra MATLAB – skriptene. Før kalibreringsrutinen ble knyttet til databasen laget derfor studenten med ansvar for kalibreringsprosessen [16] en C++ klasse med metoder som i GetPtInfo (se 6.1.1). Man måtte dessuten gjøre et oppslag i metadatafilen for å kunne knytte saksidentifikasjonsnummeret til riktig skanner. Case Id klassen kan hente ut riktig skanner, basert på relasjonen mellom måling (Recording) og skanner i databasen.

Nye oppføringer av skannere legges til i det grafiske grensesnittet, og blir dermed uavhengig av kalibreringsprosessen. Dette gjør at man slipper å endre koden i noen klasse for hver gang man legger til en ny skanner. Det er vel verdt å merke seg at om ikke denne databaseløsningen i denne oppgaven hadde blitt laget, så ville nok skannerklassen til kalibreringsrutinen implementert at man leser innstillingene fra en XML – fil eller lignende.

8.3.3 Mulighet til å gjøre kalibreringsprosessen på andre maskiner enn serveren

En annen mulig fordel for kalibreringsprogrammet er når man kjører det mot filstruktur, er man låst til å kjøre programmet på samme maskin som filene ligger. Når kalibreringsprogrammet kjører mot databasen kan den hente filer over nettverket fra MySQL – serveren, prosessere dataene lokalt på maskinen og laste opp resultatet til serveren.

8.3.4 Oppsummering

Kommunikasjon med databasen i forhold til filstruktur har gitt mulighet for å automatisere arbeid som måtte gjøres manuelt før. Databaseklassen som kommuniserer med bildetabellen i databasen har hos meg lagret bildene som en QFile variabel. Dette er fordi man skal kunne håndtere bilder som er lagret som 16 bits en kanals tiff – bilde. Denne bildetypen er noe uvanlig, og ble ikke støttet av Qt sin klasse for håndtering av bilder, QImage. Bruken av QFile i kalibreringsprogrammet ga en advarsel hver eneste gang det ble opprettet en ny fil uten filsti. Siden det gjennom en typisk gjennomkjøring av kalibreringsprogrammet opprettes mange nye bilder, valgte studenten som har utviklet kalibreringsprogrammet [16] å tilpasse bildeklassen til å bruke QByteArray istedenfor QFile. På skannerklassen valgte han å fjerne noe funksjonalitet som ikke var relevant for bruk i kalibreringsprogrammet.

9 Oppsummering og videre arbeid

9.1 Oppsummering

Gjennom dette prosjektet er det blitt laget en databaseløsning som skal avløse dagens lagringsmetode i filstruktur hos Balter Medical. Identifiseringen av attributter og attributtnavn var en tidskrevende prosess. Det hendte ofte at det ble identifisert nye etter jeg trodde vi hadde funnet en endelig databaseløsning. Den endelige versjonen av databasesystemet har vist seg å fungere bra for de programmene som benytter den i dag.

For å få lastet inn data i databasen er det blitt laget et opplastingsprogram. Opplastingsprogrammet har en automatisert del som laster opp data som er koblet sammen i en tekstfil kalt metadata. Det er også en brukerstyrt del, der man laster opp data som ikke er spesifisert bra nok i metadatafilen.

Klassen som har ansvar for å laste opp data spesifisert i metadatafil fungerer bra i dag. For bedre oversikt i koden og for å gjøre videre utviding av opplastingsprogrammet enklere, burde den være delt opp i flere klasser med spesifiserte arbeidsområder. Dette ville bli gjort om det hadde vært mer tid.

En del av oppgaven var også å lage en klasse som kan hente ut og sette inn nye data for et saksidentifikasjonsnummer. Det nåværende formål med klassen er den skal implementeres av kalibreringsrutinen utviklet av en annen student.

På samme tid som min oppgave har det jobbet to andre studenter som har tatt i bruk databasen. Ene studenten har jobbet med kalibreringsprogramvare og tok i bruk klassen som er beskrevet over, den andre utviklet visualiseringsprogramvare. Begge har implementert databasestøtte med suksess.

Samlet at både kalibrerings- og visualiseringsprogrammet har kunnet ta i bruk databasen der tolkers som at designvalgene som ble tatt rundt struktureringen har vært bra.

Arbeidet med dette prosjektet har gjort at jeg har satt meg inn i et fagområde innen fysikk som ligger langt utenfor det jeg har vært borti tidligere. Arbeidet har vært spennende og lærerikt. Utviklingen har vært i samarbeid med personer med annen faglig bakgrunn. Dette har vært en utfordring som jeg har lært mye av.

9.2 Videre arbeid

For databasen vil det i fremtiden bli aktuelt å legge til fysiologi og morfologidata. Det kan bli aktuelt å legge til muligheter for å legge dem til ved hjelp av opplastingsprogrammet som er laget, men det kan bli mer naturlig å automatisere denne prosessen, slik som det ble gjort med kalibreringsrutinen.

Båndbredde og varians som ble fortalt om i 8.1.2 er naturlig utvidelse av databasen.

Dagens løsning har bare mulighet for å laste opp klinisk protokoll. Det er ikke mulighet for å endre eller slette. Når en klinisk protokoll erstattes av en annen, sier bestemmelser fra helsemyndighetene at den gamle skal gjøres utilgjengelig for de fleste i selskapet. Det er krav om at det skal være vanskelig å få tilgang på utdaterte kliniske protokoller [2]. Dette kan løses ved at den kliniske protokollen slettes fra databasen, eller at tilgangen til klinisk protokoll skal styres ved hjelp av tilgangsrettigheter.

Det er også lagt til attributter i databasen der det kan lagres hvem som har validert at det er skikkelig kvalitet på pasientbildene, og datoen de ble validert. Det er per dags dato ikke mulighet for å legge til disse. Først må det identifiseres om dette er noe det kunne vært aktuelt å gjøre før man laster opp data og legge til hvem som har validert og datoen det ble gjort i metadatafilen. Siden dette vil knytte systemet mer opp mot den gamle løsningen mot filstruktur tror jeg heller at det kunne vært fornuftig å ha et eget valideringsprogram.

For fremtidige utvidelser mener jeg at det vil være viktig å skille opplastingsrutinene med annen programfunksjonalitet. Dette er for at man skal unngå feil i databasen.

I fremtiden bør det lages et program separert fra opplastingsprogrammet som gir mulighet for validering av bildesett, å gjøre foreldete kliniske protokoller utilgjengelig. I dette programmet kan man også legge til muligheten for å skrive rapporter om pasienter, lesjonene de har og diagnosen de har fått.

Bibliografi

1. **Kreftforeningen.** [Internett] [Sitert: 23 Mars 2008.] http://www.kreftforeningen.no/vp/multimedia/archive/00000/Brosjyre__Pdf___F_flek_408a.pdf.

2. **Nilsen, Kristian P.** Personal Comment. 2008.

3. **Kreftregisteret.** Hudkreft. [Internett] 17 August 2007. [Sitert: 23 Mars 2008.] http://kreftregisteret.no/fakta/foflekkreft.htm.

4. **Pasienthåndboka.** [Internett] Kreftforeningen, 7 Februar 2007. [Sitert: 23 Mars 2008.] http://www.pasienthandboka.no/default.asp?mode=document&documentid=1806.

5. **Malmin, Simen.** Personal Comment. 2008.

6. **K. P. Nielsen, L. Zhao, E. R. Sommersten, J. J. Stamnes, G. A. Ryzhikov and M. S. Biryulina, K. Stamnes, J. Moan.** Retrieval of the physiological state of human skin from UV-VIS reflectance spectra - A feasibility study. *submitted to Journal of photochemistry and photobiology B: Biology.* 2008.

7. **MD, David L. Swanson.** Newsletter. *International Society for Digital Imaging of the Skin.* [Internett] 31 Januar 2008. [Sitert: 06 Mars 2008.] http://i-s-d-i-s.com/ISDIS_News_08v2.pdf.

8. **American Academy of Dermatology.** publikasjoner. *American Academy of Dermatology.* [Internett] 2007. [Sitert: 6 Mars 2008.] www.aad.org/public/publications/pamphlets/sun_malignant.html.

9. **Sommerville.** *Software Engineering 8.* s.l. : Pearson Education Limited, 2007. ss. 119-126.

10. **Trolltech.** About Qt. [Internett] [Sitert: 24 Mars 2008.] http://doc.trolltech.com/4.3/aboutqt.html.

11. **caBIG.** About caBIG. [Internett] National Cancer Institute, 13 Mars 2008. [Sitert: 24 Mars 2008.] https://cabig.nci.nih.gov/overview.

12. **MySQL.** About MySQL. [Internett] [Sitert: 23 Mars 2008.] http://www.mysql.com/about/.

13. **MySQL.** MySQL Query Browser. [Internett] [Sitert: 24 Mars 2008.] http://www.mysql.com/products/tools/query-browser/.

14. **MySQL.** Referansemanual. [Internett] [Sitert: 21 Mars 2008.] http://dev.mysql.com/doc/refman/5.0/en/adding-users.html.

15. **Langeland, Jan Martin.** *Visualisering av optimaliserte bilder av hudsykdommer.* Bergen : s.n., 2008.

16. **Vikshåland, Svein Even.** *Optimalisering av bildebehandling for diagnose av hudsykdommer.* Bergen : s.n., 2008.

17. **Raghu Ramakrishnan, Johannes Gehrke.** *Database management systems.* New York : McGraw - Hill, 2003. ISBN 0-07-246563-8.

18. **Martin, Robert Cecil.** *Agile Software Development Principles, Patterns, and Practices.* s.l. : Pearson Education, Inc., 2003. ISBN 0-13-597444-5.

Appendiks Begrepsliste

Biopsi. En biopsi er når man fjerner en vevsprøve som skal undersøkes under mikroskop.

Biopsy [Engelsk], se biopsi.

Clinical Diagnosis [Engelsk], se klinisk diagnose.

Clinical Investigation Plan (CIP) [Engelsk], se klinisk protokoll.

Calibration Matrix [Engelsk], se kalibreringsmatrise.

Calibration Standard (CS) [Engelsk], er bilder tatt mot helt hvit bakgrunn. Disse bilddene blir brukt for å kalibrere pasientbildene så disse er uavhengige av eksponeringstid og lysfordelingen til lampene.

Collection [Engelsk], se samling.

Dark Current (DC) [Engelsk], se mørkestrømsbilder.

Dark Current 2 (DC2) [Engelsk], se mørkestrømsbilder 2.

Exposure Time [Engelsk], se lukketid.

Fysiologi er i denne oppgaven fysiologiske parametre knyttet til en lesjon.

Fysiology [Engelsk], se fysiologi.

Kalibreringsbilder er en fellesbetegnelse for både kalibreringsstandardbilder og mørkestrømsbilder. Grunnen til fellesbetegnelsen er at de brukes til å kalibrere pasientbilder.

Kalibreringsmatrise er en matrise med matriser laget av samlinger med kalibreringsstandard- og mørkestrømsbilder

Kalibreringsstandard blir omtalt som CS. Er bilder tatt mot et hvitt motiv som har egenskapen at den har nær 99% reflektans.

Kalibrerte bilder pasientbilder (level 0) som har blitt kalibrert ved hjelp av CS og DC bilder. De nyeste skannere bruker også DC2 bilder.

Klinisk diagnose er legens diagnose ved å visuelt inspisere føflekken.

Klinisk protokoll er retningslinjer for medisinske forsøk.

Lesjon område av huden med uregelmessigheter som skal undersøkes.

Level 0 er ukalibrerte pasientbilder. Det er dette formatet pasientbildene har når de blir tatt av skanneren.

Level 1 er kalibrerte pasientbilder.

Level 2 inneholder i dag morfologi- og fysiologidata.

Lukketid er tiden sensorene i kameraet får på å samle lys for å lage et bilde.

Malignt melanom betyr ondartet melanom. Se melanom.

Melanom er en form for hudkreft som ofte utvikler seg fra pigmentholdige celler.

Morfologi er brukt i denne oppgaven på analyse av formen på en lesjon.

Mørkestrømbilder er bilder tatt av skanneren når det ikke slippes inn lys i det lukkede skannersystemet.

Mørkestrømsbilder 2 er også mørkestrømsbilder med en annen rolle for en måling av en lesjon. Se mørkestrømsbilder.

Pathological Diagnosis [Engelsk]¸ se patologisk diagnose.

Patologisk diagnose, er diagnosen fra biopsi fra en lesjon gjort av en patolog.

Patolog lege som er spesialist i patologi.

Patologi læren om sykdommens årsaker og symptomer.

Relative Calibration Area [Engelsk], se relativt kalibreringsområde.

Relativt kalibreringsområde, er et område på skanneren som er dekket av en liten tapp. Denne tappen har bestemte egenskaper som kan brukes til å oppdage feil i kameraer eller lamper.

Råbilder, er et annet ord for level 0 bilder. Se level 0.

Samling er en tabell i databasen som samler bilder som logisk hører sammen.

Scanner [Engelsk]¸ se skanner.

Skanner¸ er verktøyet legen bruker for å ta bilder.