NTNU
Norwegian University of
Science and Technology

Mobil app for film reading device

Forfattere
Håkon Heggholmen
Christian Hådem
Even Måren Stende

Bachelor i ingeniør - data
20 ECTS
Institutt for Datateknologi og Informatikk
Norges teknisk-naturvitenskapelige universitet,

16.05.2018

Veiledere
Prof. Marius Pedersen
Prof. Sony George

Sammendrag av Bacheloroppgaven

Tittel:	**Mobil app for film reading device**
Dato:	16.05.2018
Deltakere:	Håkon Heggholmen Christian Hådem Even Måren Stende
Veiledere:	Prof. Marius Pedersen Prof. Sony George
Oppdragsgiver:	Piql AS
Kontaktperson:	Håkon G. Larsson, hakon.larsson@piql.com
Nøkkelord:	Piql, Bildebehandling, Film, Applikasjon, Kamera, Mobil, LaTeX, Java, Android, NTNU
Antall sider:	105
Antall vedlegg:	9
Tilgjengelighet:	Åpen

Sammendrag:

Piql AS har utviklet en langtidslagringsteknologi ved å lagre data binært på 35mm fotosensitiv film. For å vise frem produktets robusthet og enkelhet, ønsker Piql å se på muligheter for å lese inn data ved bruk av en smarttelefon. Hovedoppgaven vår har vært å utvikle en applikasjon som prosesserer et bilde av slik film, for deretter å dekode de binære dataene slik at en bruker kan se innholdet som er lagret. Applikasjonen er utviklet til Android 5.0+, kjører primært i Java og benytter Piql sitt eksisterende bibliotek for å dekode de binære dataene.

Summary of Graduate Project

Title:	**Mobile app for film reading device**
Date:	16.05.2018
Authors:	Håkon Heggholmen Christian Hådem Even Måren Stende
Supervisors:	Prof. Marius Pedersen Prof. Sony George
Employer:	Piql AS
Contact Person:	Håkon G. Larsson, hakon.larsson@piql.com
Keywords:	Piql, Imageprocessing, Film, Application, Camera, Mobile, LaTeX, Java, Android, NTNU
Pages:	105
Attachments:	9
Availability:	Open

Abstract: Piql AS has developed a long-term storage technology by storing binary data on photosensitive film. To showcase the products robustness and simplicity, Piql is looking at opportunities by reading film data with smartphones. Our main task has been to develop an application that processes a picture of such film, followed by decoding the binary data to a human readable format and present it to the user. The application is designed for Android 5.0+, runs primarily in Java, and uses Piql's existing library to decode the binary data.

Forord

Piql har gitt oss en interessant og unik problemstilling som det har vært spennende å være med å løse. Vi håper at produktet som er utviklet kommer godt til nytte hos Piql.

Vi vil takke Håkon G. Larsson for å ha vært en motiverende og lett tilgjengelig oppdragsgiver. Oppgaven har vært interessant og utfordrende i tillegg til å ha vært veldig åpen og gitt oss mye frihet rundt utvikling.

Vi vil og takke veilederene våre prof. Marius Pedersen og prof. Sony George for å jevnlig ha hjulpet oss med å gi konstruktiv tilbakemelding og faglig innspill gjennom hele prosjektet.

Vi ønsker også å takke prof. Ivar Farup for å ha hjulpet til med å diskutere problemstillinger og teori rundt bildebehandlingen.

Innhold

Figurer

Tabeller

1 Innledning

1.1 Problemområde

Piql AS er et norsk selskap med over 30 kontorer rundt om i verden som utvikler teknologi og løsninger innen langtidslagring av digitale data. Selskapets kjernekompetanse er innen overføring av digitale filer fra og til fotosensitiv film [1]. Kundene til Piql er foreløpig store institusjoner som f.eks. arkiver og biblioteker som har viktig informasjon og historie som de ønsker å ta sikker backup av. For å oppnå sikker langtidslagring og arkivering av data, så har Piql tatt i bruk fotosensitiv film utviklet for å vare i over 500 år. Med et proprietært datalagringsformat som Piql selv har utviklet, kan data printes binært på film og leses av ved hjelp av spesialisert hardware (Piql Writer™ og Piql Reader™).

Problemet i dag er at det kan være vanskelig for de ansatte hos Piql å formidle til potensielle kunder hvordan teknologien bak deres lagringstjenester fungerer. Med utgangspunkt i denne problemstillingen skal prosjektgruppen utvikle en mobil applikasjon for Piql. Applikasjonen skal kunne brukes som et verktøy for å demonstrere teknologien bak deres produkt og sikre seg potensielle kunder.

Hovedoppgavene til en slik applikasjon vil være å digitalisere filmdata ved bruk av mobilkamera, dekode data fra Piql sitt proprietære datalagringsformat og presentere det på et menneskelig leselig format. Digitalisering av filmdata innebærer utvikling og bruk av flere bildebehandlings-algoritmer slik at bildet kan pålitelig dekodes. Grunnet begrensede ressurser begrenses omfanget av bildebehandlingen. Det må også tas hensyn til smarttelefoners begrensede ytelse. De binære data, som vil være lagret på film, vil dekodes med et programvare bibliotek utviklet av Piql.

1.2 Oppgavedefinisjon

Det skal utvikles en mobilapplikasjon for digitalisering av binærdata lagret på film. Dataene skal også dekodes og presenteres på menneskelesbart format.

For å digitalisere filmdata skal brukeren kunne holde mobiltelefonen over en dataramme, hvor data er lagret binært på Piql sitt proprietære format. Applikasjonen skal gjenkjenne rammen med kameraet, lese ut data og presentere innholdet. Piql har et eksisterende programvarebibliotek for å dekode binære data fra det ferdig prosesserte bildet. I tillegg fikk vi utlevert utstyr til å lese film, og et budsjett på 8000kr til innkjøp av en mobiltelefon og makro-linse til formålet (beskrevet i Seksjon 3.3).

1.3 Formål

Målet med oppgaven er å utvikle en mobil applikasjon som kan lese filer fra eksisterende film. Metoder og prinsipper for bildebehandling er sentrale i oppgaven, samt Android utvikling og kamerateknologi. Prosjektet vil gi gruppemedlemmene erfaring med reelle problemstillinger og forbereder derfor til arbeidslivet.

Resultatmål

- Lage en mobil applikasjon som ansatte hos Piql kan bruke til å hente ut og vise filer fra en filmrute/dataramme som bruker Piql sitt proprietære format.

Effektmål

- Demonstrere uavhengighet av eksisterende spesialisert hardware for fremtidig uthenting av data.
- Bygge tillit til teknologien bak lagringstjenester på film, og dermed øke salg.

1.4 Målgruppe

Rapporten og produktet har hver sine målgrupper. Produktet er for Piql, og rapporten til akademisk bruk.

1.4.1 Målgruppe for produktet

Piql sine lagringstjenester er først og fremst ment for store institusjoner som arkiver og biblioteker. Man ønsker å øke tilliten til teknologien og lagringstjenester utviklet av Piql. Applikasjonen skal primært brukes av Piql ansatte for å demonstrere teknologien bak produktet.

1.4.2 Målgruppe for rapporten

Rapporten er for interesserte som ønsker å se på vår gjennomføring av bacheloroppgaven, og faglig interesserte i bildebehandling eller Android applikasjonsutvikling. Rapporten skal inneholde tilstrekkelig informasjon, slik at en leser kan i teorien reprodusere våre resultater.

1.5 Rammer

Prosjektet vil naturlig bestå av flere ressurser som vil begrense prosjektets størrelse og omfang. Kriterier som tid, lover, teknologi og økonomi setter rammer for prosjektet.

Tidsrammer

Tid er den mest avgjørende ressursen i prosjektet. Prosjektet startet 10. Januar og den endelige rapporten leveres innen 16. Mai.

Økonomiske rammer

Budjett er satt til 8000kr, dette er finansiert av Piql. Budsjettet dekker innkjøp av smarttelefon og makro-linse (beskrevet i Seksjon 3.3). Utstyret overleveres til Piql ved avslutning av prosjektet.

Juridiske rammer

For å kunne utføre prosjektet var det nødvendig med innsyn i proprietær kildekode. Konfidensialitetsavtale (se Vedlegg H) ble inngått for å ivareta Piql sine interesser. På grunn av konfidensialitetsavtalen blir ikke endringer utført i Piql biblioteket beskrevet dersom koden kreves for kontekst. Enerett på den utviklede kildekoden i prosjektet tilfaller Piql (se Vedlegg G). Det blir benyttet to tredjeparts biblioteker i prosjektet, med lisenskrav, disse er omtalt i Kapittel 5.

1.6 Utviklingsmodell

For å ha en god struktur på utviklingsprosessen ble det valgt en systemutviklingsmodell som skal følges under utviklingsperioden. Det ble satt opp en oversikt over egenskaper ved prosjektet, og undersøkt hvilke av de mest vanlige modellene som passer best.

1.6.1 Hovedtrekk ved vårt prosjekt

- Under utviklingen vil det være mye avhengighet mellom funksjonalitet i deler av programmet. Dette medfører at mange moduler i programmet må fullføres i en bestemt rekkefølge.
- Det er lite krav om oppfølgning og verifisering under utvikling fra Piql sin side. Dette kan eventuelt erstattes med verifisering fra veiledere.
- På grunn av rapporten som skal skrives, må beslutninger under utvikling dokumenteres.
- Fordi oppgaven hadde lite krav til funksjonalitet må funksjonalitetskravene justeres etter egne forutsetninger.

1.6.2 Begrunnelse for valg av modell

Plandrevne utviklingsmodeller inneholder mye dokumentasjon og planlegging, og er best egnet for store prosjekter. Smidige utviklingsmodeller egner seg derimot bedre for mindre prosjekter, hvor kunden vil være med på å forme produktet og komme med innspill. Smidige utviklingsmodeller tillater også at krav blir endret underveis.

Ettersom det kan oppstå endringer under utviklingen, vil det ikke være mulig å lage en fast plan og kravspesifikasjon for hele applikasjonen. Det er derfor valgt å gå bort fra **Fossefall** og **RUP** (**R**ational **U**nified **P**rocess) siden disse baserer seg på å gjøre all planlegging i starten, før implementering.

Inkrementell utviklingsmodell passer ikke, da Piql ikke ønsker å ta imot delleveranser av applikasjonen for å teste den og gi tilbakemelding. I tillegg vil det bli tidkrevende å planlegge og sende ut delversjoner.

Smidige utviklingsmodeller har mange gode egenskaper relevant for prosjektet. I **XP** så kan f.eks. refactoring (omstrukturere kode uten å endre funksjonalitet) og parprogrammering være nyttige på enkelte viktige moduler i applikasjonen. Utviklingsmodellen har lite krav til dokumentasjon, som kan bli vanskelig siden produktet skal kunne videreutvikles av Piql, og det skal skrives en rapport av prosjektet.

Fordelen med **Scrum** er at den innfører en logg med oppgaver (Scrum backlog). Backloggen kan være nyttig om det dukker opp flere ønsker til funksjonalitet underveis. Oppdragsgiver ønsket ikke å fylle rollen som ”product owner” (produkt eieren som setter krav).

Kanban tar i bruk en enkel tavle hvor arbeidet blir delt opp og organisert. Med dette blir det mindre veksling av arbeidsoppgaver og man får god oversikt på hva som må gjøres.

Scrum ble valgt da det tilsynelatende passer godt i denne typen prosjektarbeid.

1.6.3 Roller og detaljer i utviklingsmodellen

Scrum har en rekke med roller som må angis i prosjektet. Scrum master vil være Håkon Heggholmen, siden han også har prosjektlederrollen (se Seksjon 1.7.1) er det naturlig at han også tar denne rollen, men han vil også være utvikler. Product owner vil være oppdragsgiver Håkon G. Larsson, men det vil derimot ikke være mye oppfølging av produktet under utviklingen etter ønske ifra oppdragsgiver. Varigheten på hver sprint ble satt til to uker. På slutten av en sprint vil det være en ”Scrum retrospect” møte hvor sprinten vil bli vurdert og dokumentert. På starten av hver Sprint utføres en ”Sprint planning meeting” for å planlegge de neste 2 ukene.

1.7 Prosjektorganisering

1.7.1 Roller og ansvarsforhold

Det ble valgt å ha en prosjektleder under prosjektet etter råd gitt i forelesning om bacheloroppgaver. En prosjektleder er en person som tar store beslutninger i prosjektet og passer på at prosjektet går etter planen. Håkon Heggholmen er valgt som prosjektleder, og har ansvar for kommunikasjon med oppdragsgiver Håkon G. Larsson, seniorutvikler hos Piql AS.

Christian Hådem er valgt som dokumentasjonsansvarlig. Dette innebærer ansvar for føring av timelogger og dokumentasjon av beslutninger som gjøres underveis. Even Måren Stende er utstyrsansvarlig og skal passe på at alt utstyr blir levert tilbake til Piql etter prosjektperioden. Grunnet få deltagere vil alle i gruppen fungere som utviklere i tillegg til sine tildelte roller.

Veilederene er Marius Pedersen og Sony George som begge har doktorgrad innenfor bildebehandling og har flere publikasjoner innenfor samme område. Marius Pedersen er også leder for Norsk laboratorium for farge og visuell prosessering.

1.7.2 Gruppens bakgrunn

Gruppen består av tre fulltidsstudenter som studerer dataingeniør på NTNU i Gjøvik. Gruppen har erfaring med C, C++, SQL, Python, og enkel PHP og x86 Assembly. Håkon Heggholmen og Christian Hådem har erfaring med Java og applikasjonsutvikling. Even Måren Stende har hatt programvaresikkerhet og Christian Hådem har i tillegg hatt matematikk 3. Gruppen har også hatt systemutvikling og er kjent med utviklingsmodeller og arkitekturmodeller. Andre relevante kunnskaper er algoritmiske metoder, lineær algebra og enkel bildebehandling.

Oppgaven krever god forståelse innenfor bildebehandling og applikasjonsutvikling. For å utføre oppgaven må gruppen anskaffe manglende kompetanse, spesielt innenfor bildebehandling og Android applikasjonsutvikling.

1.7.3 Rutiner

Det er satt som mål at hver person skal jobbe 30 timer i uken med prosjektet. For å garantere dette ble det satt opp en timeplan for prosjektarbeid. Gruppen skal møte opp på avtalte steder minst 4 dager i uken (mandag, tirsdag, torsdag, fredag), med mindre annet er avtalt.

Gruppen skal også ha ukentlige møter hvor det diskuteres hva som er gjort og hva som må gjøres. Disse møtene skal det skrives korte referat av, og en Trello tavle skal oppdateres med eventuelle nye oppgaver.

1.7.4 Grupperegler

For å få et godt samarbeid på gruppen med god effektivitet er det satt opp grupperegler. Disse skal hjelpe til med å avgjøre interne problemer som måtte oppstå under prosjektet-perioden.

Fravær

Alle gruppemedlemmer møter til avtalt tid, fravær skal meldes til gruppen på forhånd. Ved lengre fravær (over en uke kombinert umeldt/ubegrunnet fravær) skal veileder kontaktes for å diskutere situasjonen.

Konflikter

Ved uenigheter av implementasjon og mindre beslutninger rundt prosjektet skal hele gruppen inkluderes i diskusjonen om uenigheten. Dette gjelder kun beslutninger som ikke har innvirkning på prosjektet som helhet, f.eks. grafiske design-valg og andre reversible valg.

Ved uenigheter av større beslutninger/vedtak i prosjektet skal et møte settes opp internt i gruppen hvor alle partene får tid til å komme med egne argumenter. Disse møtene blir dokumentert sammen med motargumenter. Løser ikke dette konflikten blir veileder kontaktet.

1.8 Om rapporten

Bakgrunnsteori beskriver enkelte relevante områder mer detaljert for de som ikke er kjent med disse emnene. Hvis noen av disse emnene er kjent område kan de ignoreres. Kravspesifikasjon og design kapitlene vil gå igjennom hvordan applikasjonen er bygd opp i fra et bruker- og utviklerperspektiv. Implementasjonen er delt opp inn i tre deler: Bildebehandling, Android applikasjonsutvikling og Importering av biblioteker. Her blir det gjennomgått problemer, løsninger og fremgangsmåter ved implementasjon av de forskjellige funksjonalitetene i applikasjonen. Til slutt i Kapittel 8 vises det hvordan applikasjonen ble testet, og i Kapittel 9 diskuteres det hvordan prosjektet ble utført.

1.8.1 Terminologi

For å forstå innholdet i rapporten må enkelte ord og begreper forklares og settes i kontekst:

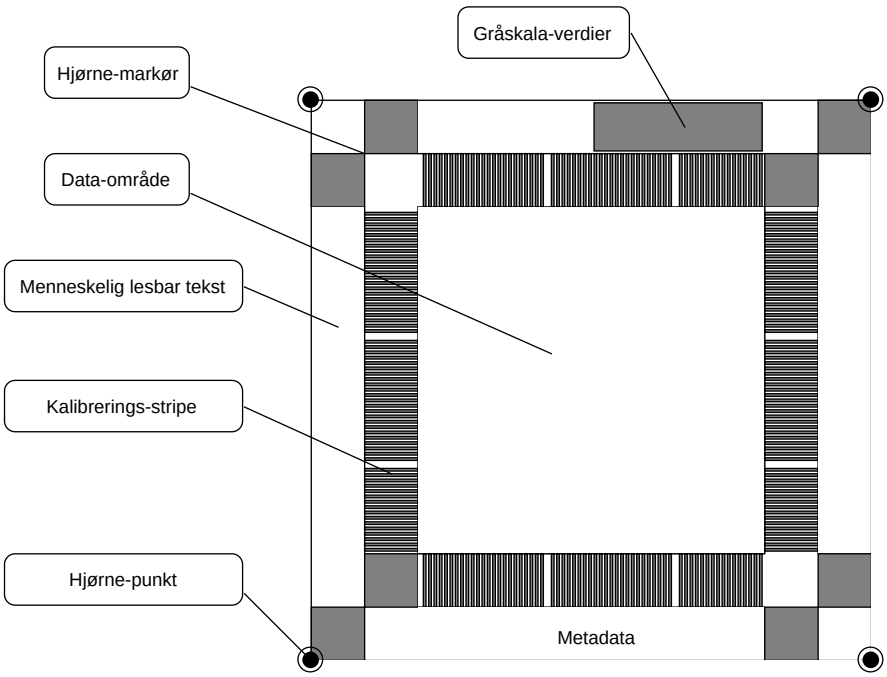
- App og applikasjon, sammen med bøyningene, vil bety det samme gjennom hele rapporten.
- Mobil, smarttelefon, telefon og mobiltelefon betyr det samme.
- **Filmreader** er navnet til applikasjonen.
- **GUI** står for **G**raphical **U**ser **I**nterface som betyr grafisk brukergrensesnitt. Grafisk brukergrensesnitt er det brukeren ser og samhandler med for å bruke applikasjonen.
- **MP** står for **M**ega**P**iksler og beskriver hvor mange millioner elementer det er i en kamera sensor.
- **JNI** står for **J**ava **N**ative **I**nterface som brukes for å kalle på funksjoner skrevet i C fra Java. For registrering/eksportering av funksjoner i C brukes JNIexport. Funksjonen som skrives for å kalle på den eksporterte funksjonen i Java er en native funksjon.
- **IDE** står for **I**ntegrated **D**evelopment **E**nvironment og er et program som brukes til å utvikle programvare.
- **Overlay**: Informasjon som blir tegnet over et bilde for å gi brukeren visuell tilbakemelding, uten å påvirke bildet under prosesseringen.
- **FPS**: Antall bilder hvert sekund som blir vist på skjermen (**F**rames **P**er **S**econd).
- **SVN**: Subversion er en eldre teknologi for organisering av kildekode (repository), som ligner mye på en synkronisert nettverksmappe.
- **Kontur**: Ordet kontur brukes for å beskrive en sammenhengende kant mellom punktkoordinater i et bilde. Synonymt med omriss.
- **Sampling**: Sampling/prøvetakning brukes som et uttrykk for å hente ut piksler fra deler av et bilde.
- **Samplings-rate**: Samplings-rate brukes for å beskrive hvor nøyaktig et bilde måler selve innholdet i en dataramme.
- **ROI (Region of interest)**: ROI/Interesseområdet er et kvadratisk område i bildet som ramme-søk begrenses til for å forbedre ytelsen.
- **Kalibrering og kalibrasjon**: Ordet kalibrering brukes for å beskrive prosessen for å generere kalibrasjon. Kalibrasjon er resultatet av kalibreringen som består av en 3x3 kamera-matrise og 5 forvrengningskoeffisienter.

2 Bakgrunnsteori

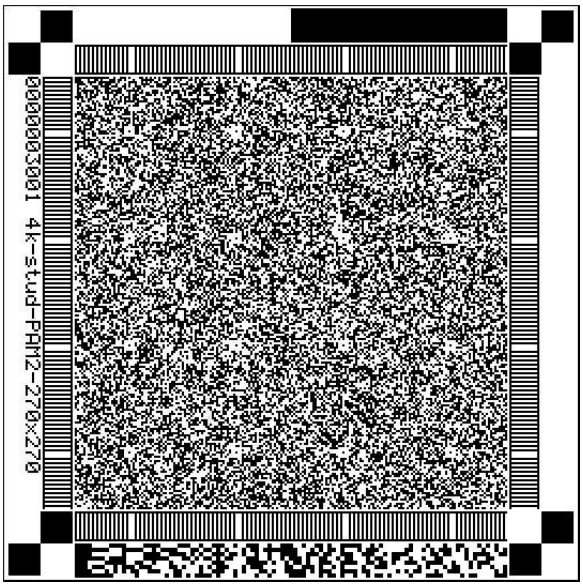
For å lese og tolke resten av rapporten kreves det kunnskap om det faglige innholdet. Dette kapittelet er delt opp etter tema og beskriver kort teori som kreves for å forstå rapporten.

2.1 Piqls proprietære format

For å kunne lagre data på film har Piql utviklet sitt eget format for å lagre binære data. Piql har et dokument hvor det i større detalj er beskrevet hvordan formatet er bygd opp, i tillegg til hvordan data lagres binært [2].



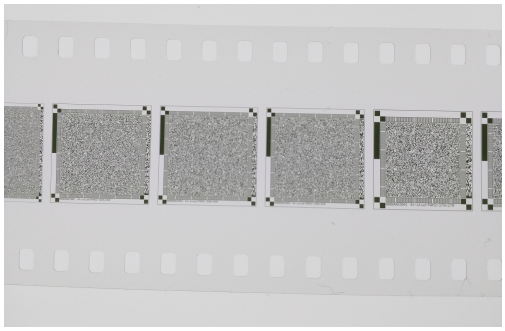
Figur 1: Uttrykk brukt til å beskrive ulike deler av data-rammen



Figur 2: Eksempel på en filmramme

Beskrivelse av filmrammer (se Figur 1 og 2):

- På utkanten av rammen er det en tynn stripe rundt hele området. All data relatert til rammen holdes innenfor denne linjen for å markere én enkel ramme.
- I hvert av hjørnene er det 2x2 svart-hvitt ruter i et sjakkbrett mønster (også kalt hjørnemarkører). Piql sitt bibliotek benytter disse hjørnene for å finne rammen.
- Området mellom de to øverste hjørnemarkørene, viser hvor mange gråskala-verdier som er brukt for å lagre data i rammen. I Figur 2 benyttes det kun to gråskala-verdier.
- Rektangelet på venstre side inneholder menneskelig lesbar tekst. Teksten blir brukt til å lagre kort informasjon for å lettere kunne skille rammer før de er dekodet og brukes *ikke* i dekodingen.
- Nederst på rammen er ligger metadata til rammen.
- I midten av rammen ligger dataområdet hvor fildata er lagret.
- Rundt dataområdet er det 4 kalibrerings-striper (også kalt referansebånd). Referansebånd brukes til å spore synkroniseringspunktene og hvor tett dataene er lagret.



Figur 3: Flere rammer på 35mm film

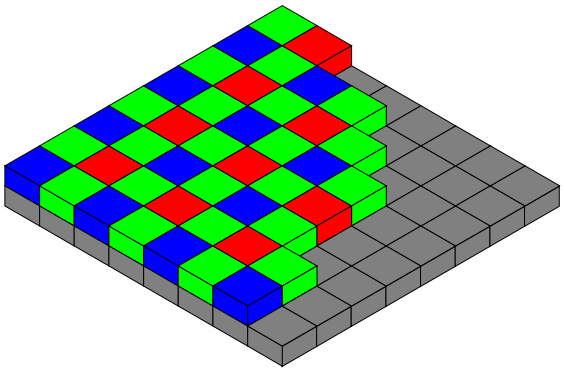
Rammer blir printet ut på 35mm film som vist på Figur 3. Flere gråskala-verdier og høyere oppløsning på dataområdet krever høyere samplings-rate og jevn belysning i bildet. I våres tilfelle vil datarammen kun ha to gråskala-verdier (svart og hvitt), og 270x270 oppløsning, som gir oss en begrensning på 4KB lagringsplass. Hvilken som helst fil kan lagres på en dataramme. For å kunne lagre flere filer i rammen, pakkes innholdet inn som en ".tar" fil [3].

2.2 Bildebehandling

For å forstå kapittelet om bildebehandling kreves det kunnskap om hvordan et kamera produserer et bilde, og hvordan forvrengninger kan oppstå.

2.2.1 Kamera-matrise

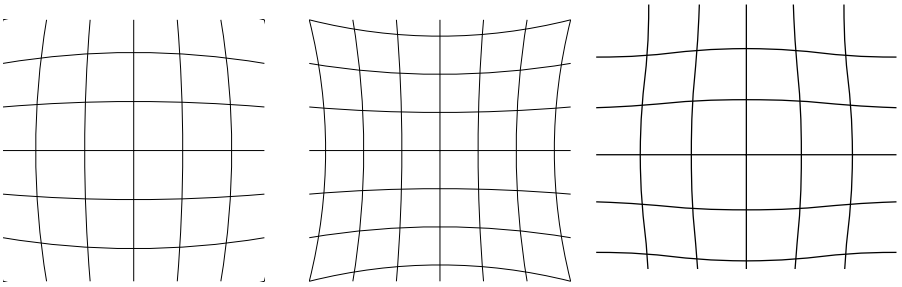
For å ta et bilde bruker mobilkamera en linse for å fokusere lyset inn på en sensor-matrise. Antall sensorelementer i matrisen beskrives i MP. For å kunne ta farge-bilder så ligger det et farge-filter i et bayer-mønster over matrisen (se Figur 4). Et digitalt bilde lages ved å lese lysstyrken på hvert sensorelement, deretter interpoleres bildet for å få en rød grønn og blå fargeverdi for hvert piksel.



Figur 4: Fargefiltre i bayer mønster over kamera sensor matrise.
Bildet er hentet fra Wikipedia [4]

2.2.2 Radiell forvrengning

Forvrengninger i bildet oppstår når lyset treffer kamera-sensoren på ulike måter. Radielle forvrengninger skjer avhengig av formen på linsen. De mest vanlige formene for radielle forvrengninger er barreling, pincushion og mustache (se Figur 5).



(a) Illustrasjon av barreling (b) Illustrasjon av pincushion (c) Illustrasjon av mustache forvrengning

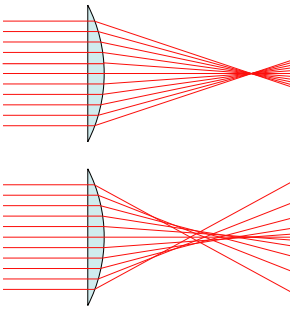
Figur 5: Illustrasjoner av ulike former for radielle forvrengninger.
Figurene er hentet fra Wikipedia [5]

2.2.3 Tangentiell forvrengning

Tangentiale forvrengninger forekommer når linsen ikke er parallell med, eller er sentrert over kamerasensoren. Forvrengningene forekommer på lik måte som radiell forvrengning, men førskjøvet/rotert og ikke symmetrisk.

2.2.4 Sfærisk aberrasjon

Sfærisk aberrasjon forekommer når linsen ikke fokuserer lyset på et punkt (se Figur 6). Resultatet av dette er at bildet blir uklart langs kanten av bildet (se Figur 16 for eksempel).



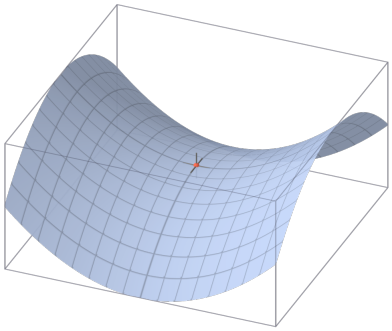
Figur 6: Illustrasjon av linse og lys med og uten sfærisk aberrasjon.
Bildet er hentet fra Wikipedia [6]

2.2.5 Kromatisk aberrasjon

På samme måte som sfærisk aberrasjon fokuseres lyset ujevnt, men ulikt for bølgelengder slik at lyset langs kantene av bildet separeres i likhet med prismer.

Figur 7: Eksempel på et bilde med og uten kromatisk aberrasjon.
Bildet er hentet fra Wikipedia. [7]

2.2.6 Sadel-punkt



(a) Graf med sadelpunkt plottet i tre dimensjoner.
Bildet er hentet fra Wikipedia. [8]

(b) Eksempel på hjørne-markør med sadelpunkt

Figur 8: Bilder med sadel-punkter.

Et sadelpunkt er punktet i midten av en graf som ser ut som en sadel. Punktet ligger der den deriverte er lik null, som ikke er et maksimal- eller et minimal punkt (se Figur 8 (a)). Figur 8 (b) viser hjørne-markørene på en dataramme hvor sadelpunktet ligger i midten der de svarte kvadratene møtes.

2.3 OpenCV

Med all bildebehandlingen nødvendig i applikasjon vil vi ta i bruk biblioteket OpenCV [9]. OpenCV er et kraftig open source bibliotek for bildebehandling og maskinlæring som kan brukes i bl.a. Android.

2.4 Android begreper

Under utviklingen av en Android applikasjon brukes det flere begreper som kan være ukjent for leseren. Disse begrepene skrives ofte på engelsk, men vi oversetter begrepene til norsk der det er hensiktsmessig.

2.4.1 Aktivitet/Activity

En aktivitet i Android er et ”vindu” hvor det kan plasseres flere GUI elementer. For eksempel kan en aktivitet bestå av flere bilder, knapper eller andre GUI elementer. Aktiviteter byttes ut med andre aktiviteter fortløpende som gir en oversiktlig og brukervennlig stuktur i applikasjonen [10].

2.4.2 Visning/View

Visninger er de såkalte GUI elementene som brukes i aktiviteter. Eksempler på visninger er en ImageView som kan vise bilder, eller ImageButton som er en knapp med et ikon som brukeren kan trykke på. En visning kan kobles til flere handlinger (events) for å f.eks. endre på et ikon eller bilde når brukeren trykker et sted på skjermen [11].

2.5 Make systemer

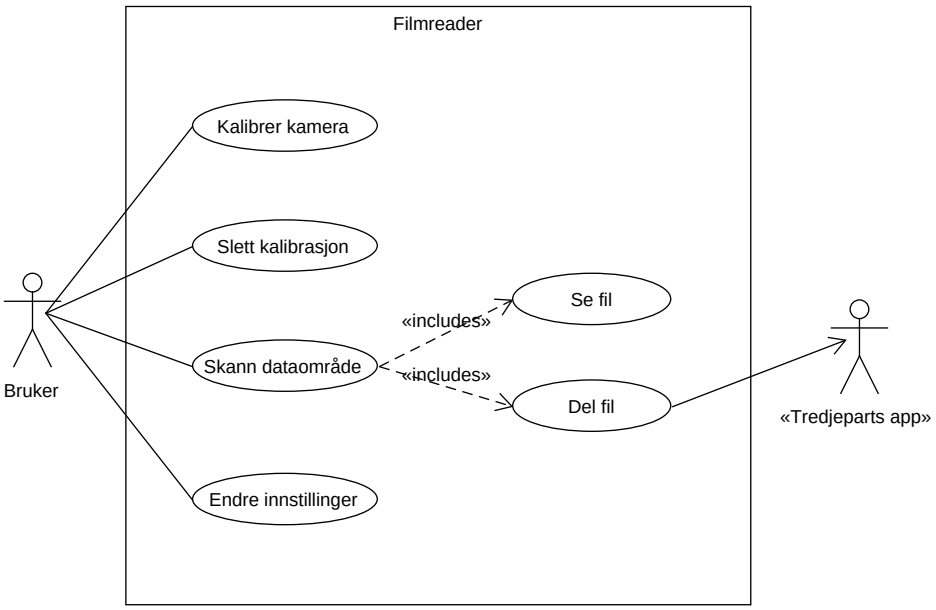
Make systemer, som Qmake og Cmake, er verktøy som genererer ”make-filer”. Disse filene inneholder instrukser om hvordan et prosjekt skal kompileres og hvordan kodefilene skal linkes sammen til én eksekverbar fil. I prosjekter med mange filer vil Make systemer redusere tiden det tar å kompilere filene, siden det bare er filene som er endret på som må kompileres på nytt.

3 Kravspesifikasjon

For å få en bedre forståelse om hva som skal utvikles ble det satt opp en oversikt over oppdragsgiverens ønsker om funksjonalitet, krav og kvalitetsønsker.

3.1 Use case diagram

Vi opprettet et use case diagram for å få oversikt over alle brukerinteraksjonene i applikasjonen.



Figur 9: Use case diagram

3.1.1 Aktører

Aktørene beskrevet i use-case diagrammet, som vist på Figur 9 er:

- **Bruker:** En person som bruker applikasjonen har tilgang til alle funksjonene. Siden applikasjonen kun skal brukes lokalt av én person inneholder den ingen form for adgangskontroll.
- **Tredjeparts applikasjon:** En annen applikasjon på samme enhet som kan brukes til å vise filformater som ikke støttes av denne applikasjonen, f.eks. PDF filer.

3.1.2 Høynivå use case beskrivelse

Use case	Kalibrer kamera
Aktører	Bruker
Hensikt	Rette opp forvrengninger
Beskrivelse	Brukeren tar flere bilder fra flere vinkler, slik at applikasjonen kan bruke dette til å rette opp utvalgte forvrengninger i bildet.

Use case	Slett kalibrasjon
Aktører	Bruker
Hensikt	Fjerne den aktive kalibrasjonen
Beskrivelse	Applikasjonen fjerner den aktive kalibreringen slik at brukeren kan kalibrere på nytt.

Use case	Endre innstillinger
Aktører	Bruker
Hensikt	Endre på innstillinger
Beskrivelse	Brukeren kan endre innstillinger. Følgende innstillinger kan endres på: • Kalibrering (på/av) • Preview type • Oppløsning • Kalibrering størrelse • Antall kalibrerings bilder

3.1.3 Detaljert beskrivelse av use case

Vi har valgt å beskrive det mest omfattende use caset (skann dataområde) i større detalj, for å få med alternative hendelser og feilsituasjoner.

Use case	Scann dataområde, Se fil, Del fil
Aktører	Bruker, «Tredjeparts app»
Hensikt	Scanne et dataområde for å lese ut data
Forbetingelser	Brukeren har navigert seg til hoved aktiviteten og kameraet er kalibrert eller kalibrering er slått av
Etterbetingelser	Brukeren kan se innholdet i dataområdet hvis denne eller en tredjeparts applikasjon støtter formatet
Normal hendelsesflyt	1. Brukeren holder kameraet over et dataområde. 2. Applikasjonen gjenkjenner et dataområde. 3. Dataområdet rettes opp, beskjæres og roteres slik at det er klart til dekoding. 4. Dataområdet blir sendt til Piql sitt bibliotek for dekoding. 5. Alle de dekodede filene settes opp i en liste, hvor brukeren selv velger hvilken fil de vil se. 6. Innholdet i filen vises.
Variasjoner og feilsituasjoner	2. Applikasjonen gjennkjenner et område som ikke er en ramme i det hele tatt (falsk-positiv). Applikasjonen fortsetter til forhånds-prosesseringen uten en gyldig ramme. 3. Forhånds-prosesseringen gjenkjenner ikke rotasjon og bildet blir ikke rotert før det skal dekodes. Applikasjonen fortsetter til dekoding med et ødelagt bilde. 4. Dekodingen feiler. Kvaliteten til bildet er ikke god nok eller formatet støttes ikke. Bildet forkastes og følgende steg blir ikke utført. 6. Filformatet støttes ikke av applikasjonen (noe annet enn PNG, JPG eller tekst). Brukeren får muligheten til å åpne filen i en ekstern applikasjon som er installert på mobilen.

3.2 Krav til systemet

3.2.1 Funksjonelle krav

Oppdragsgiver hadde veldig lite konkrete krav til applikasjonen, men flere generelle kvalitetsønsker rundt funksjonalitet. På grunn av manglende krav, så satt vi opp våre egne krav for å øke kvaliteten på det endelige produktet.

Piql sine krav

- Dekodingen av den binære data skal gjøres av Piql sitt bibliotek.
- Applikasjonen skal vise data som blir lest fra film.

Gruppen sine krav

- Et dataområde skal oppdages i sanntid.
- Applikasjonen skal under demonstrasjon ha mindre enn tre steg/trykk for å hente ut data fra film.

3.2.2 Tekniske krav

Ved valg av hvilken platform vi skulle utvikle applikasjonen på, sto det hovedsaklig mellom Android og iOS. Før vi bestemte oss diskuterte vi hvilken platform vi ønsket å utvikle applikasjonen på. Vi valgte Android på grunn av følgende argumenter:

- Alle på gruppen inkludert oppdragsgiver har Android smarttelefon og bruker dette daglig.
- iOS utvikling bruker Swift som ingen av oss er kjent med, men Android utvikling bruker Java som to av oss er kjent med.

3.2.3 Krav om brukergrensesnitt

Det er ikke satt noen krav til brukergrensesnittet i applikasjonen. Google har klare designprinsipper og mange ressurser tilgjengelig, som viser hvordan de mener applikasjoner bør være utformet [12]. Fordi applikasjonen kun skal brukes til demonstrasjoner, så vil vi prioritere funksjonalitet over brukergrensesnittet. Mange av visningene, som f.eks. ListView som viser data grafisk i en liste, bruker allerede Google sine retningslinjer som standard.

3.2.4 Kvalitetsønsker

Ytelsen under dekoding av et dataområde kan være for dårlig for praktisk bruk av applikasjonen, spesielt på eldre smarttelefoner. Vi har satt som mål at prosesseringstiden skal den holdes så lav som mulig for å forbedre brukervennligheten. Generelt i funksjonalitet og design vil applikasjonen ligne på en QR kode app. Applikasjonen vil hente inspirasjon fra følgende trekk fra QR kode lesere:

- Kameravisningen er synlig når applikasjonen starter.
- Ser etter datarammer i sanntid.
- Innholdet blir presentert når datarammen er funnet.

Applikasjonen bør også være robust ettersom den skal brukes for å demonstrere pålideligheten til teknologien bak Piql sine lagringstjenester. Android smarttelefoner har stor variasjon i maskinvare som den kjøres på, hvor den mest relevante forskjellen for vårt prosjekt er kameraet. Når applikasjonen skal vises frem, eller kunder skal prøve ut applikasjonen selv er det viktig at den ikke kræsjer. Vi har derfor lagt ekstra vekt på robusthet i applikasjonen.

3.3 Utstyr

I planleggingsfasen fikk vi et budsjett på 8000kr fra oppdragsgiver for innkjøp av en smarttelefon og en makrolinse.

3.3.1 Mobil

Smarttelefonen skulle brukes til å teste applikasjonen. Ved valget av smarttelefon ble følgende vurdert:

- Antall MP i kameraet for best mulig sampling-rate av datarammen.
- Prosessorkraft for raskest mulig prosessering av bildet i sanntid.
- Pris innenfor det gitte budsjettet på 8000kr.
- Støtte for uthenting av ukomprimerte bilder via Android SDK.

- Støttet Android versjon for å kunne kjøre applikasjonen.

For å finne et utvalg av telefoner som oppfyller kriteriene og kan bestilles i Norge brukte vi Google [13] og Prisjakt [14]. Deretter brukte vi nettsiden GSMArena [15] for å sammenligne spesifikasjonene til de ulike mobilene. Til slutt endte vi opp med å gå for Samsung Galaxy A8 (2018) som akkurat hadde blitt lansert. Denne mobilen har åtte kjerner totalt i prosessoren, 2 kjerner med 2.2 GHz og 6 kjerner med 1.6 GHz klokkehastighet. Smarttelefonen har 4 GB med RAM, 32 GB lagringsplass og et 16 MP kamera.

3.3.2 Linse

Siden datarammen befinner seg på 35mm film, så må vi bruke en makrolinse for å forstørre dataområdet. Valg av makrolinse går under samme budsjett og hadde flere utfordringer:

- Lite marked og dermed lite tilgjengelighet.
- Ingen intern erfaring rundt makrolinser for mobil.
- Varierende kvalitet som er vanskelig å se uten å ha kjøpt produktet.

På grunn av utfordringene så forventet vi å måtte teste flere makrolinser. Linsen vi valgte måtte bestilles fra utlandet. Da vi fortalte oppdragsgiver om dette sa han at linsen ikke var noe bra, så han valgte ut en annen linse for oss. Det var ikke nødvendig å teste flere makrolinser, så linsen vi endte opp med var ”Black Eye Smart Phone Lens 3 in 1” (Figur 10).

3.3.3 Oppsummering av utstyr

1. Februar var vi på besøk hos Piql AS sitt hovedkontor i Drammen hvor vi fikk utdelt smarttelefonen vi hadde valgt ut, i tillegg til følgende utstyr:

- Hama 8x linse, som ble brukt før vi fikk Black eye linsen.
- 35mm film med testrammer for å teste applikasjonen.
- Lomography smartphone film scanner.
- KAISER slimlite LED.

Etter noen få tester viste det seg a Lomography sin smartphone film scanner ikke var det vi trengte. Smarttelefonen sammen med linsen passet ikke inn i Lomography skanneren. Den 26. Februar fikk vi tilsendt Black Eye linsen. Vi fikk i tillegg to ekstra filmruller, hvor begge inneholdt testdata og kalibrerings-mønstre. Black Eye linsen passet heller ikke i Lomography skanneren så vi brukte ikke skanneren.

Når vi fikk lest ut data fra datarammene, viste det seg at data til én fil var lagret over flere datarammer. Vi kontaktet oppdragsgiver angående filoppdelingen og det var ikke meningen at vi skulle få slike filmruller. Den 13. April fikk vi en ny filmrull hvor hver dataramme inneholdt kun én fil.

Til slutt endte vi opp med følgende utstyr:

Figur 10: Alt utstyr som vi fikk tildelt fra oppdragsgiver

Punktene under beskriver utstyret i Figur 10:

1. Samsung Galaxy A8 (2018).
2. Hama 8x linse.
3. Black Eye Smart Phone Lens 3 in 1.
4. Lomography smartphone film scanner.
5. KAISER slimlite LED.
6. 3 filmruller som inneholder datarammer og kalibreringsdata.
7. 35mm film som inneholder datarammer

4 Design

4.1 Systemarkitektur

Valg av systemarkitektur er viktig fordi det påvirker ytelsen, robusthet, distribuering og vedlikeholdbarhet i et system [16]. For systemarkitekturen i vårt prosjekt har vi valgt å bruke pipe & filter modellen.

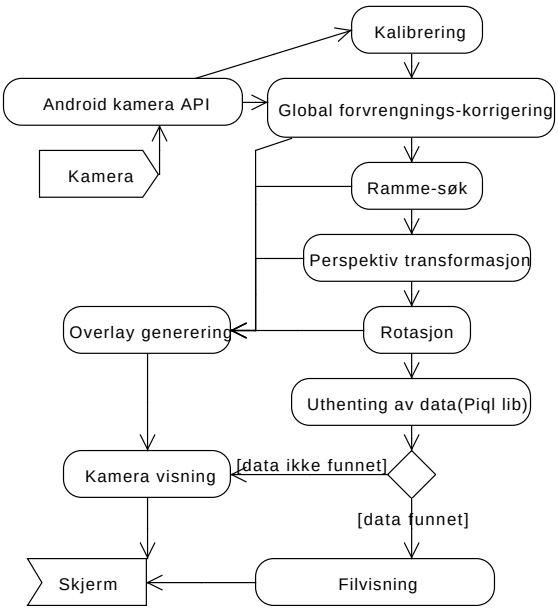
4.1.1 Begrunnelse for valg av systemarkitektur

Applikasjonen har to aktører og kommuniserer ikke med andre enheter. Vi kan derfor utelukke systemarkitekturer som har fokus på å tilrettelegge dette. Applikasjonen består også for det meste av komponenter som tar informasjon inn, prosesserer den og gir ulik output. Fordi Pipe-and-filter hadde tilsynelatende mange fordeler for bruk i prosjektet, hvor andre systemarkitekturer var designet for å tilrettelegge kommunikasjon, ble Pipe-and-filter valgt.

Pipe-and-filter legger til rette for:

- Kontinuerlig prosessering av data fra kameraet.
- Uavhengighet mellom moduler som kan kobles sammen på ulike måter, f.eks ramme-søk, rotasjon og punktgjenkjenning i dataområdet. Det kan derimot ikke settes en hvilken som helst sammenkobling av modulene, fordi datatypene som kreves som input i modulene varierer.

4.1.2 Detaljert beskrivelse av systemarkitektur



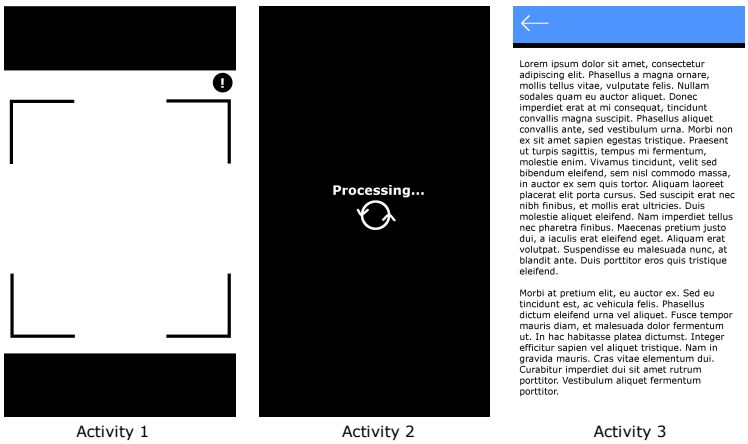
Figur 11: Pipe & Filter diagram

Figur 11 viser hvordan informasjonsflyten relatert til prosessering av bilder under vanlig bruk vil forekomme. Figuren viser bare informasjonsflyten på et overordnet plan, hvor senere kapitler går mer i detalj. Enkelte funksjonaliteter som ikke er en del av den normale program-flyten (fil-deling, programinnstillinger, GUI) er ikke med i figuren, men er beskrevet i senere kapitler. Beskrivelse av Figur 11:

- **Kamera:** Kamera på telefonen.
- **Android kamera API:** All funksjonalitet relatert til å hente ut bilder fra kamera, og klargjøre det for prosessering.
- **Kalibrering:** Bilder av et kalibrerings-mønster, som brukes til å finne ut hvor mye bildet er forvrengt og for å generere kalibrasjon.
- **Global forvrengnings-korrigering:** Ved bruk av kalibrasjons-data korrigeres forvrengninger i bildet skapt av linsen.
- **Ramme søk:** Finner de fire hjørnepunktene ytterst i rammen.
- **Perspektiv transformasjon:** Transformerer perspektivet i bildet om til et todimensjonalt plan.
- **Rotasjon:** Finner ut hvilken vei rammen er rotert og roterer bildet slik Piql-biblioteket krever.
- **Overlay generering:** Overlay brukes til å kunne tegne over bildet med tekst og annen tilbakemelding til brukeren under prosessering, uten å påvirke selve bildet som blir prosessert.
- **Uthenting av data:** Programvarebibliotek skrevet av Piql som konverterer rammebilder til fil.
- **Filvisning:** Visning i Android applikasjonen som inneholder filer som ble lest fra rammen.
- **Kamera visning:** Kamera visning som viser prosesserte bilder.

4.2 Plan for brukergrensesnitt

For å få en brukervennlig applikasjon lagde vi en plan tidlig i prosjektet på hvordan vi ønsket at brukergrensesnittet skulle se ut. Ingen på gruppen hadde erfaring med brukergrensesnitt design, men vi visste at det var lurt å holde det enkelt og ryddig.



Figur 12: Alle planlagte aktiviteter i applikasjonen

Planen var å holde brukeren på kjent område. Siden applikasjonen primært bruker kamera, åpnes applikasjonen rett i en kameravisning som vist på Activity 1 i Figur 12. Kameravisningen utvides til den tar hele skjermen, eller så langt det går i én dimensjon uten å ødelegge størrelsesforholdet (aspect ratio) til bildet. Det et overlay på kameravisningen for å gi brukeren visuell tilbakemelding under prosesseringen. Det var planlagt å legge en informasjonsknapp øverst til høyre i denne aktiviteten for å informere brukeren om bruk av applikasjonen.

Etter en dataramme har blitt funnet vil dekodingen av et dataområde starte. Under dekodingen vil brukeren få tilbakemelding på at applikasjonen jobber som vist i Activity 2 på Figur 12.

Etter prosesseringen er ferdig vil den dekodede dataen vises, sammen med en knapp som viser at brukeren kan gå tilbake og scanne mer data (Aktivitet 3 på Figur 12).

5 Importering av biblioteker

5.1 Apache Commons Compress

For å kunne lese ”tar” filer valgte vi å finne et tredjeparts bibliotek siden vi ikke hadde kompetanse eller tid til å implementere dette selv. Vi endte opp med å bruke Apache Commons Compress [17]. Den hadde Apache lisens som vi var nødt til å forholde oss til, samtidig som vi var nødt til å legge til vår egen Apache lisens i prosjektet [18]. For å importere biblioteket trengte vi kun å legge til noen linjer i ”build.gradle” filen så lastet Android Studio ned biblioteket automatisk.

5.2 OpenCV

Vi valgte å bruke OpenCV (Open Source Computer Vision Library) som er et kraftig bibliotek for bildebehandling og gjenkjenning (se Kapittel 2.3). OpenCV bruker en BSD lisens som i hovedsak sier at hvis vi skal redistribuere må vi ha med lisensen [19]. Dette er ikke tilfellet i vår applikasjon, så vi behøver ikke å tenke på å ta med lisens for OpenCV. For å importere OpenCV biblioteket gjorde vi følgende:

1. Lastet ned den siste versjonen av biblioteket (”Android pack” versjon 3.4.0) fra OpenCV sin hjemmeside [20].
2. Pakket ut zip filen og importerte modulen i Android studio.
3. Den statiske delen av biblioteket kopieres inn i prosjektet som kreves av OpenCV (”OpenCV-android-sdk/sdk/native/libs” til prosjektets ”app/src/main/jniLibs”).
4. Vi satt riktig Android versjon i ”build.gradle” filen til biblioteket (minimum versjon 21, target og compile versjon 26). For begrunnelse av valgt Android versjon se Kapittel 7.1.1.

5.3 Piql sitt bibliotek

Piql har et eksisterende bibliotek skrevet i C som de bruker til å konvertere data fra og til datarammer. Dette biblioteket har vi fått tilgang til og det er planlagt at vi skal bruke biblioteket i applikasjonen. For å få tilgang til biblioteket var vi nødt til å skrive under på en konfidensialitetsavtale (Vedlegg H).

Piql sitt bilbiotek består av to hoveddeler, boxing og unboxing. Boxing er en del av biblioteket som vi *ikke* tok i bruk siden den konverterer data om til datarammer. Denne delen er ikke nyttig for oss siden applikasjonen sin jobb er å gjøre det motsatte. ”unboxing”, som vi skal bruke, tar inn et bilde av en ramme og konverterer det til data som blir pakket inn i en ”.tar” fil. Siden biblioteket er skrevet for større maskiner som Piql Reader™, så må den endres for å kunne brukes på Android.

Piql har brukt Qmake [21] for å sette opp strukturen på alle filene som er med i biblioteket deres. Qmake er et make system som bruker ”.pri” og ”.pro” filer som inneholder instrukser for hvordan biblioteket skal bygges opp. Qmake ble ikke brukt fordi den bare var delvis støttet i Android studio og hadde ikke tilstrekkelig funksjonalitet for prosjektet. Derfor ble Cmake [22] valgt. Cmake er et make system på lik linje med Qmake. Selv

om disse er veldig forskjellige så genererer de begge lignende resultater i form av en ”makefile”. I motsetning til Qmake, så har ikke Cmake noen ”.pri” eller ”.pro” filer, men bruker en tekstfil hvor alle konfigurasjonene rundt prosjektet blir lagret. Alle disse konfigurasjonene blir skrevet og lagret i en fil som heter ”CMakeList.txt”. Denne filen ligger øverst i mappe hierarkiet og inneholder informasjon om hvilke mapper og filer som skal inkluderes sammen med parametre.

5.4 Qmake til Cmake

Siden biblioteket er bygd opp med Qmake var vi nødt til å konvertere alle ”.pri” og ”.pro” filene over til en ”CMakelist.txt” fil. Å oversette fra Qmake til Cmake viste seg å være mer jobb enn det vi først hadde antatt. Ingen av oss hadde tidligere erfaring med CMake eller Qmake, så det ble det brukt mye tid på gjennomgang av dokumentasjonen, og prøving og feiling rundt konfigurasjonen. Dette førte til at vi så etter verktøy for å hjelpe oss med denne prosessen. Etter litt undersøkelse fant vi Qt Visual Studio Tools (QtVSTools) [23] og CmakeConverter [24].

QtVSTools er en modul for Visual Studio (en IDE fra Microsoft) som tar inn en ”.pro” fil og setter opp prosjektet ifølge dens konfigurasjon. For å ta i bruk QtVSTools så var vi nødt til å laste ned Qt (omfattende programvarebibliotek og verktøykasse for å kjøre kode på flere platformer) slik at vi kunne kompilere ”.pro” og ”.pri” filene. Vi hadde noen problemer med den nyeste versjonen av Qt som ikke klarte å finne kompilatoren til VisualStudio. Vi trodde først at det var noe feil med konfigurasjonen av VisualStudio. Etter litt undersøkelse fant vi ut at feilen var et kjent problem med Qt versjon 5.10. En enkel løsning for dette var å installere versjon 5.8 som ikke hadde dette problemet. Å kjøre biblioteket gjennom QtVSTools var nødvendig for at vi kunne ta i bruk CmakeConverter. CmakeConverter tar inn et Visual Studio prosjekt fil som input, og gir en ”CMakeList.txt” som output. Denne automatiske konverteringen satte opp en fil som var enkel å endre på. Etter testing og modifisering av ”CMakeList.txt”, fikk vi satt opp biblioteket i Android Studio med alle filene i Piql sitt unboxing bibliotek.

5.5 PiqlLib i Android studio

Når vi fikk satt opp biblioteket i Android Studio med de riktige konfigurasjonene, fikk vi et nytt problem. Omtrent alle filene i biblioteket hadde kode som Android Studio ikke godtok. De aller fleste feilmeldingene var fra funksjoner og variabler som en normalt finner i standard biblioteker, som f.eks. ”string.h” og ”math.h”. Det ble brukt store mengder tid på å forsøke å rette opp i disse feilene. Til slutt endte vi opp med å kontakte oppdragsgiver siden han var selv involvert i utviklingen av biblioteket og har mye erfaring med C.

Etter å ha vist oppdragsgiver hva vi hadde problemer med, kom han frem til at det var to deler av koden som måtte bli endret på. Den første delen var et tredjeparts bibliotek som hadde flere feil ifølge Android Studio. Biblioteket viste seg å ha mange filer som ikke var nødvendige. Det andre vi kom fram til var at vi måtte deklarere standard funksjonene selv i biblioteket. Siden PiqlLib er skrevet i C så er funksjoner som modifiserer minnet lov å bruke. Eksempler på slike funksjoner som Android Studio ikke godtok er memset, memcpy, memmove, strcpy og strlen. [25]. Det viste seg senere at det å deklarere disse funksjonene bare var god praksis, men ikke nødvendig. Siden vi ønsket å endre så lite som mulig eksisterende kode i Piql biblioteket ble funksjonene fjernet. Biblioteket ble konfigurert slik at Android studio ikke lengre ga noen feilmelding.

6 Bildebehandling

Det blir utført mye bildebehandling i applikasjonen for å effektivt kunne hente ut data fra filmen. Dette kapittelet omfatter bildebehandlingen hvor hvert delkapittel (med unntak av markør-deteksjon) tilsvarer en modul i Figur 11.

6.1 Kalibrering

Etter møte med veileder tidlig i prosjektet, så ble det tydelig at forvrengninger fra makro-linsen ville påvirke uthenting av data i høy grad. Det ble også utført en vitenskapelig analyse som arbeidskrav i emnet Ingeniørfaglig systememne(se Vedlegg F). Vi hadde i starten av prosjektet lest om OpenCV biblioteket som har innebygd funksjonalitet for å korrigere forvrengninger. Forvrengningene fra makro-linsen som var mest tydelig i bildet var:

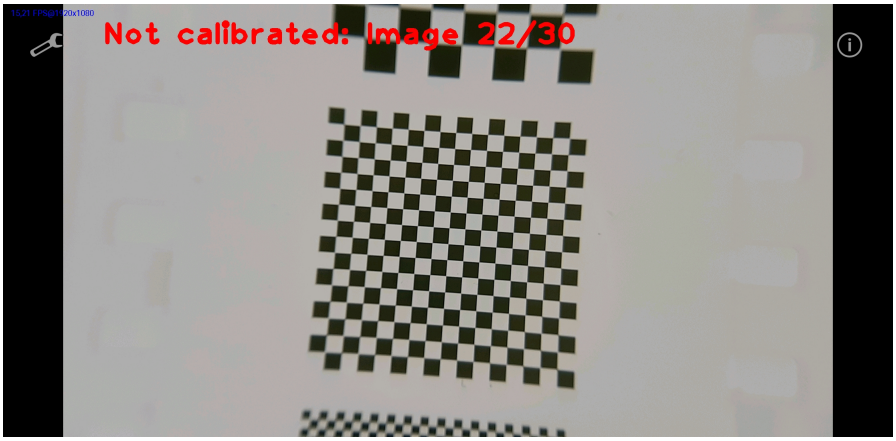
- **Kromatisk aberrasjon:** Selv om det er en tydelig forvrengning konverteres bildet til gråskala og vi så ingen nytte i å prøve å korrigere denne forvrengningen. Derfor blir ikke denne korrigert for.
- **Sfærisk aberrasjon:** Det at bildet blir ufokusert mot kanten på grunn av linsen fører til tap av informasjon i bildet. Selv om vi kunne ha forsøkt å gjøre bildet skarpere mot kanten virket ikke det hensiktsmessig.
- **Tangentiell og radiell forvrengning:** Radiell forvrengning skjer i linser med vid vinkel. Tangentiell forvrengning forskyver bildet dersom linsen ikke er sentrert over mobilkameraet. For å korrigere disse forvrengningene brukes ”undistort” funksjonen fra OpenCV, slik som beskrevet i Seksjon 6.1.2.
- **Light radience non uniformity:** Vi kunne valgt å rette opp ”light radience non uniformity”, som i tilfelle vårt betyr at det slipper mindre lys gjennom linsen langs kanten av bildet. Denne forvrengningen var ikke kritisk til applikasjonen og virket tidkrevende å implementere, så vi valgte å ikke ta den med. Ved å rette opp denne forvrengningen kan ramme-søk, rotasjon og uthenting av data fått lavere feilfrekvens.

For å korrigere forvrengninger som oppstår, så brukes Calib3d og Imgproc modulene fra OpenCV. For å ta i bruk biblioteket og legge til egen funksjonalitet, så lagde vi en kalibrasjons-klasse.

6.1.1 Fremgangsmåte

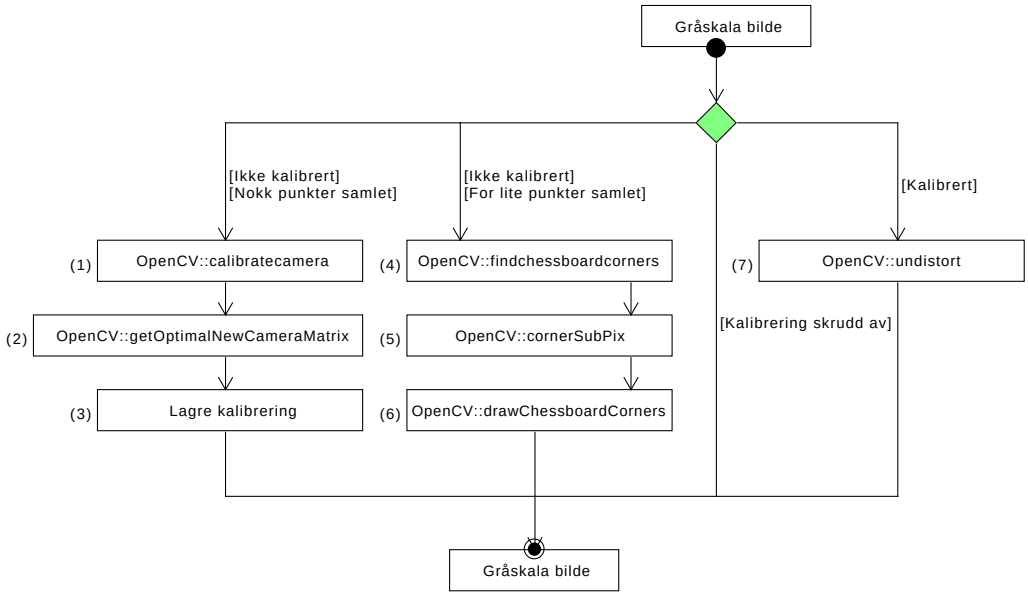
For å hente ut data fra bildet krever Piql biblioteket at bildet rettes opp, slik at den ligner mest mulig på kilde-bildet som ble skrevet ut på film. Tidlig i prosjektet undersøkte vi hva OpenCV har av funksjonalitet for dette. Før vi oppdaget calib3d modulen [26] så vi på muligheten til å bruke synkroniseringspunkter bygget inn i rammen for å rette opp bildet. Dette virket ressurskrevende å implementere med tanke på tid og manglende kompetanse.

6.1.2 Prosess



Figur 13: Forvrengt bilde av kalibrerings-mønster på 35mm film med bilde teller

Under kalibreringen scannes bilder av kalibrerings mønsteret (sjakk-mønster). Kalibreringen utføres ved at brukeren tar et bestemt antall bilder av sjakk-mønsteret som vist på Figur 13. Bildene sjekkes for sadel-punkter som blir lagret og brukt i kalibreringen. Bildene som brukes til kalibrering blir tatt automatisk med et sekund mellomrom, slik at brukeren får tid til å skifte kamera-vinkel under kalibreringen.



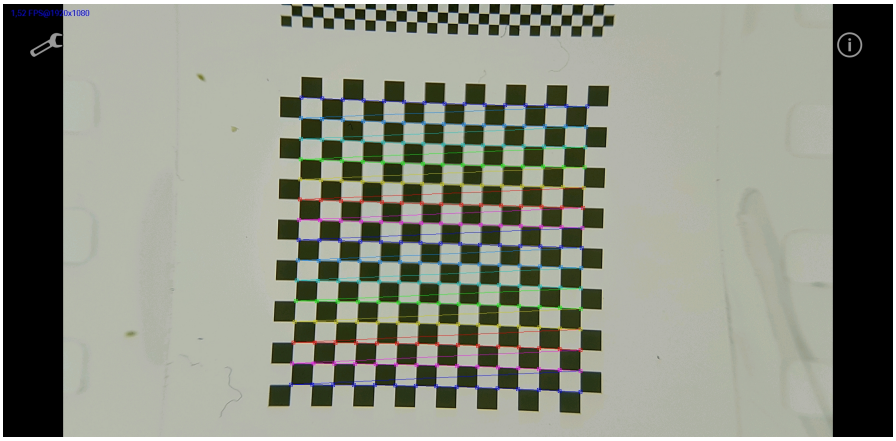
Figur 14: Flytskjema for kalibrerings prosess

Beskrivelse av prosess (punktene i Figur 14 beskrives under):

1. Etter alle punktene er funnet returneres det en 3x3 kamera matrise og 5 forvrengingskoefissienter, ved å sende inn alle punktsettene (se Figur 15) til ”calibratecamera” funksjonen fra OpenCV [26]. For mer informasjon om innholdet i kamera-

matrisen og forvrengningskoeffisientene les online dokumentasjonen [26].

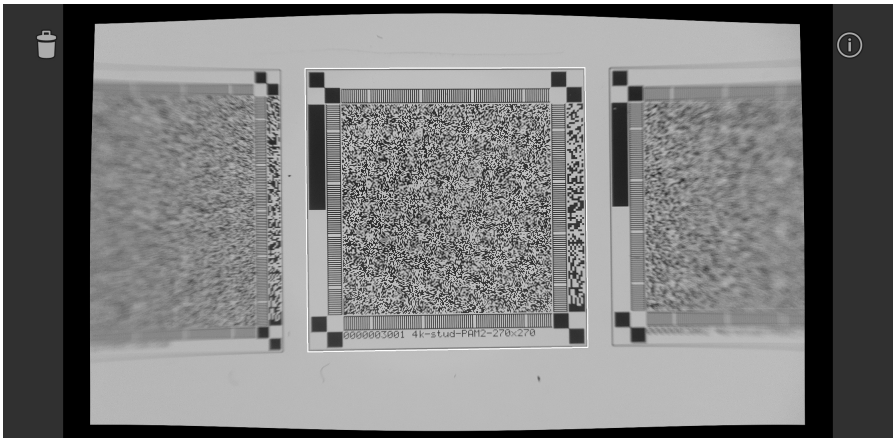
2. Funksjonen getOptimalCameraMatrix kalles med kamera-matrisen og forvrengningskoeffisientene som parameter, og returnerer et rektangulært område i bildet hvor radielle og tangentiale forvrengninger er korrigert. Dette området brukes i alle senere moduler som ROI/interesseområde (se Seksjon 1.8.1).
3. Kalibreringen lagres til fil slik at kameraet ikke må kalibreres på nytt neste gang applikasjonen kjører. Kalibreringen kan også bli slettet av bruker dersom oppløsning endres, linsen forskyves, kalibreringen var upresis eller lignende.
4. Denne funksjonen [26] fra OpenCV er brukt til å finne selve punktene i bildet. Størrelse på sjakk-mønsteret og antall bilder for kalibrering kan settes i innstillingene.
5. For å forbedre nøyaktigheten til innsamlede punkter kjøres cornerSubPix funksjonen fra OpenCV [27]. Denne funksjonen bruker kanter rundt sadelpunktet til å forbedre nøyaktigheten til punktets koordinater.
6. For å gi tilbakemelding til brukeren om hvor kalibrerings-mønsteret ble funnet tegnes punktene på bildet som vises på skjermen ved hjelp av drawChessboardCorners [26].
7. Dersom kalibreringen eksisterer (fra tidligere prosess eller på fil) kan forvrengninger i bildet rettes opp. For å gjøre dette sendes kamera-matrisen og forvrengningskoeffisientene til funksjonen OpenCV sin undistort funksjon [28].



Figur 15: Forvrengt bilde av kalibrerings-mønster med sadelpunkter funnet

6.2 Ramme-søk

For å finne ut om det eksisterer en ramme i bildet og lokalisere det for videre prosessering ble det opprettet en egen modul for dette.



Figur 16: Forvrengningskorrigert bilde med linjer mellom rammens hjørnepunkter

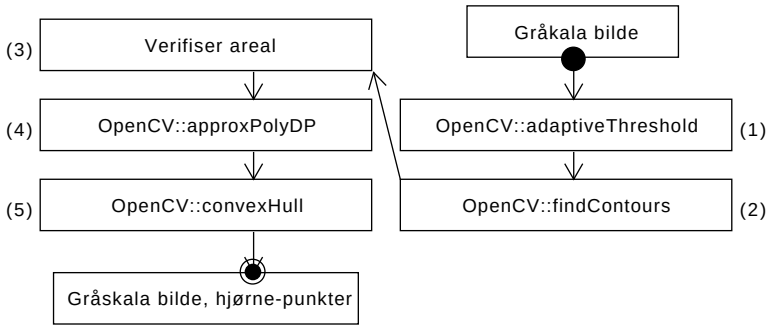
6.2.1 Fremgangsmåte

Før vi fant frem til den nåværende metoden for ramme-søk hadde vi flere mislykkede forsøk. Vi prøvde å finne rammen ved å prøve ulike metoder fra to sider av online dokumentasjonen [27] [29], og en OpenCV bok[30]. Under beskrives bare de feilede forsøkene, hvor de vellykkede er beskrevet i prosess 6.2.2.

Canny edge detection [27] ble prøvd for å finne kantene rundt rammen til bildet. Algoritmen ble ikke tatt i bruk fordi den ga kanter som hadde mange hull som var vanskelig å korrigere for.

Etter å ha funnet konturen ved hjelp av adaptiveThresholding og findCountours, så prøvde vi å implementere vår egen metode for å finne hjørnepunktene i konturen. Den tok for seg tre punkter i konturen og målte vinkelen på midtpunktet. Det så ut som punktsettet til konturen ikke var strutkurert slik vi hadde forventet. Etter litt leting etter funksjoner vi kunne utnytte, brukte vi heller approxPolyDP funksjonen fra OpenCV med høy epsilon verdi på 10 (for mer detaljert informasjon, se online dokumentasjon [29]).

6.2.2 Prosess



Figur 17: Flytskjema for ramme-søk

Beskrivelse av prosess:

1. For å finne konturer må bildet være binært. For dette brukte vi adaptiveThresholding fra OpenCV [31].
2. FindContours fra OpenCV brukes til å finne konturen (sett med punkter), som former en linje rundt rammen. Parametrene til funksjonen settes slik at den bare finner den ytterste sammenhengende konturen.
3. For å fjerne små konturer som ikke er relevante, så filtreres alle konturer hvor arealet er mindre enn interesseområdet.
4. Konturen som blir funnet inneholder mange punkter. I videre prosessering er vi bare interessert i å beholde de fire hjørne-punktene. For dette brukes approxPolyDP [29] fra OpenCV, som bare beholder punktene som utgjør en stor vinkel-endring. For mer detaljert informasjon, se online dokumentasjonen til funksjonen [29].
5. Konturen kan fortsatt inneholde artefakter, slik at vi får ”hakk” innover i konturen. For å fjerne dette filtrerer vi ut alle konkave punkter langs konturen. Om vi nå står igjen med fire punkter, så returneres de og prosessen er ferdig. For dette brukes convexHull funksjonen [29] fra OpenCV.

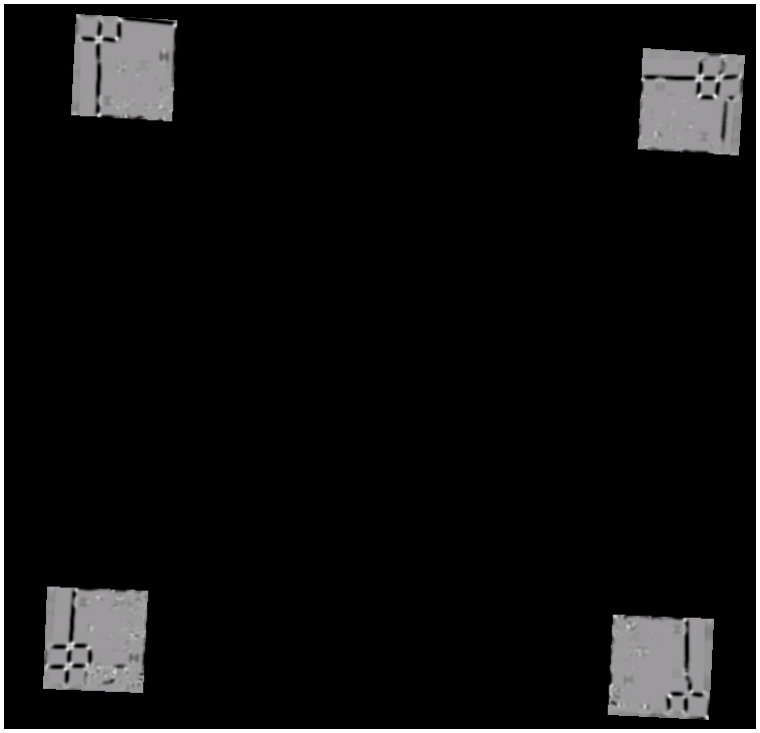
6.3 Markør-deteksjon

For å bruke Piql-biblioteket må vi først vite hvilken oppløsning rammen i bildet er. Selv om denne uferdige modulen *ikke* ble tatt i bruk, så blir den inkludert i rapporten fordi den tok mye arbeid å utvikle. Markørdeteksjon går ut på å finne midtpunktet/sadelpunktet til hjørne-markørene.

Markør-deteksjon var planlagt å bli brukt til å finne ut hvilken oppløsning rammen har. I oppgaven var det mulighet til å lese tre ulike oppløsninger: 270x270, 360x360 og 540x540.

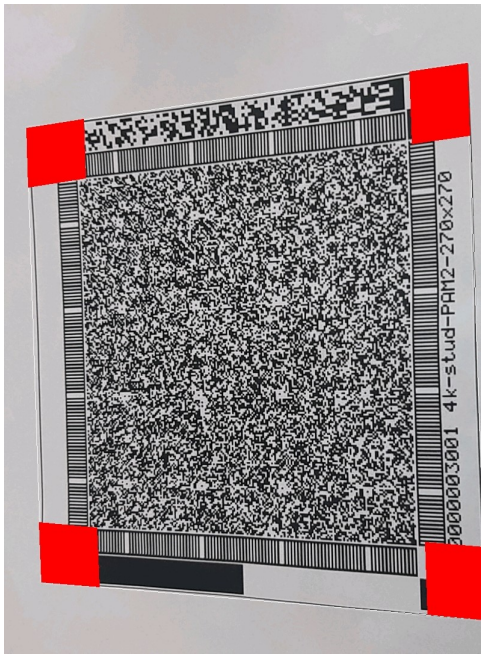
Fremgangsmåte

For å finne markørene prøvde vi først template-matching [32], Houghlines, HoughlinesP og Harris Corner Detection [27]. Template matching ga mange treff tilfeldig i bildet. Tanken var at template matching ville finne de største markørene i bildet. Det er uklart om de dårlige treffene skyldes rotasjon, forvrengninger, dårlig template eller vår implementasjon. På samme måte som template matching, hadde Houghlines og HoughlinesP lite nøyaktige resultater av ukjent årsak, selv etter mye justering av parametre. Houghlines og HoughlinesP fant ingen eller mange tilfeldige linjer i bildet, hvor målet var å søke etter linjene mellom midten av hjørne-markørene i rammen.

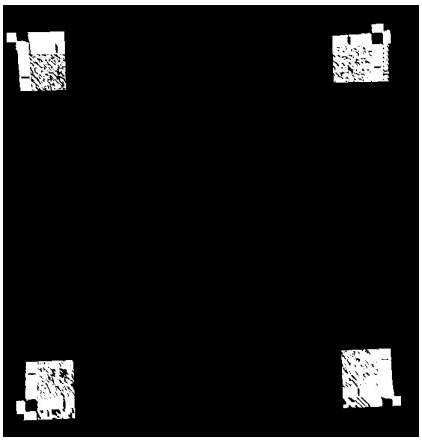


Figur 18: Bilde av forsøk på bruk av Harris Corner Detection

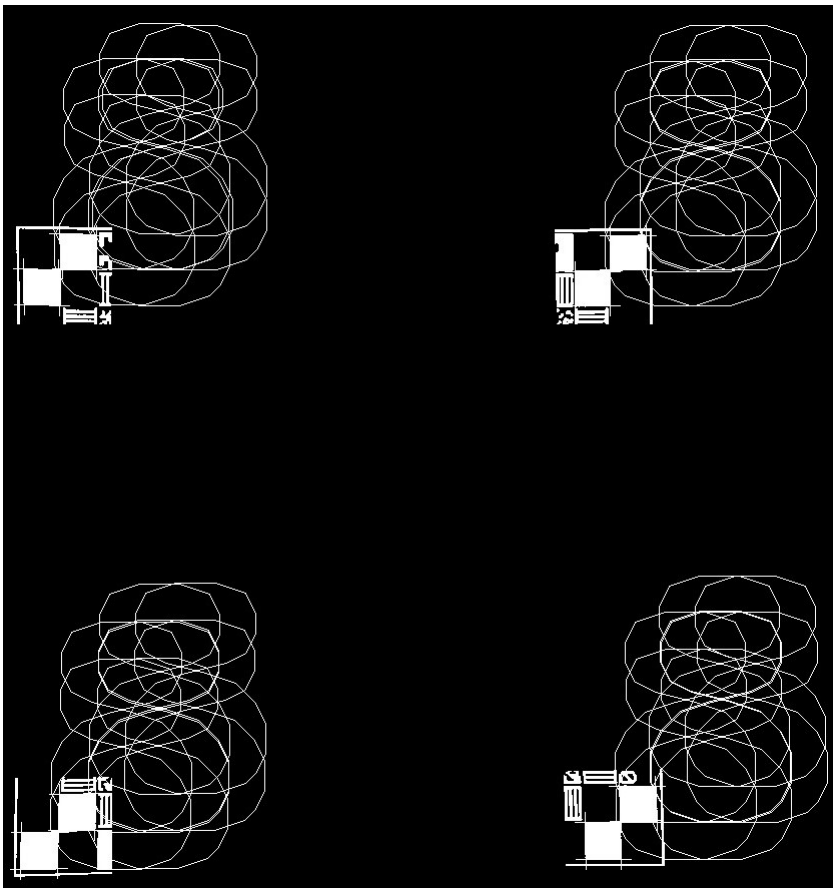
Harris Corner Detection (som vist på Figur 18) var en CPU krevende operasjon hvor det var vanskelig å filtrere ut riktig resultat.



(a) 270x270 Ramme på papir med maskerte hjørner



(b) Binær 540x540 ramme med auto-justerende maske størrelse



(c) 270x270 ramme med fire 8-punkts konturer

Figur 19: Bilder av forsøk på markør-deteksjon

Maske

Markør deteksjon viste seg å være veldig CPU-krevende. For å ekskludere punktene rundt rammen og lengre inne i rammen (dataområdet) så ble det laget en maske for relevant område, som på Figur 19 (a). Masken blir laget som et område mellom fire punkter for hvert hjørne. Selve hjørnepunktet av rammen, to vektorer fra hjørnepunktet som peker på nabohjørnene ganget med koeffisient, og et midtpunkt som konstrueres ved å summere vektorene og hjørnepunktet. Denne metoden ble gjenbrukt, modifisert og beskrevet i høyere detalj i Seksjon 6.5. Til sammen konstrueres 16 punkter (4 for hvert hjørne), som blir brukt til å lage masken.

Det mest vellykkede forsøket til å starte med var å justere maskestørrelsen til det ble funnet fire konturer med 8 punkter. Om det ikke ble funnet noen konturer utvides maskestørrelsen innover i bildet, og fire konturer hentet på lik måte som i ramme-søk i Figur 19 (b) (findContours, approxPolyDP).

Det ble mye prøving og feiling for å finne hjørnepunktene. Det ble vanskelig å separere hjørne-markørene fra resten av rammen på de høyere oppløsningene i Figur 19 (a) på grunn av forvrengninger i bildet.

Ettersom makro-linsen ikke hadde kommet enda og det gikk mye tid til å justere maske-søkeren til høyere oppløsninger, så ble ferdigstillingen av markør deteksjon modulen utsatt.

Chess metoden

Etter linsen kom innså vi at nåværende metoder ikke var nøyaktige (ved visuell inspeksjon), og startet å se etter litteratur på området. Vi fant en publikasjon [33] som også hadde et implementasjonseksempel skrevet i C, som vi skrev om til Java. Vi endte opp med å ikke ta i bruk markør-deteksjon på grunn av manglende tid og algoritmen var implementert med en fast 5 piksel radius.

Ved å ikke ferdigutvikle denne modulen medfører det en begrensning i den ferdige applikasjonen. Applikasjonen leser derfor bare data fra rammer med 270x270 oppløsning fordi oppløsning på rammen ikke kan gjenkjennes.

6.4 Perspektiv transformasjon

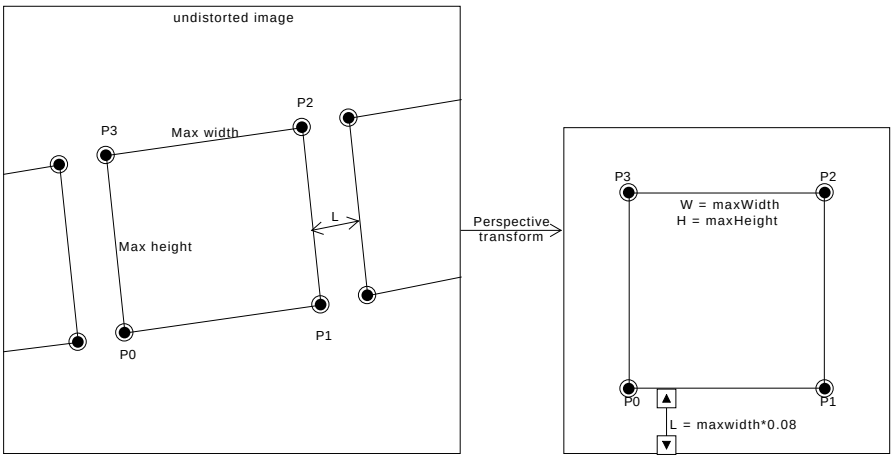
For å hente ut data fra bildet krever Piql biblioteket at gråskala-verdien til hvert piksel i bildet er invertert. Bildet må også være uten perspektiv (flatt) med marg (som vist i Figur 20 og 22).

Siden bildet av rammen ikke blir tatt rett ovenfra og ofte med rotasjon, så må det utføres en perspektiv-transformasjon før prosesseringen kan fortsette.

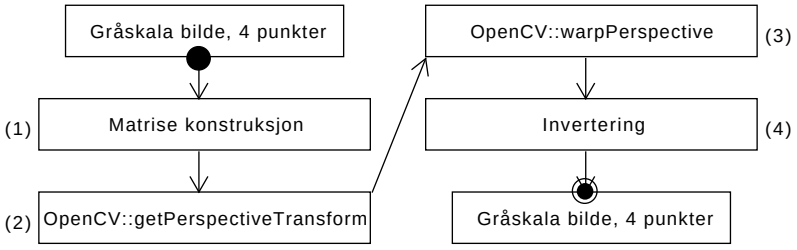
6.4.1 Fremgangsmåte

Etter rammen var funnet og biblioteket var integrert, så kom vi frem til at perspektiv transformasjon var nødvendig. Dette fant vi ut under møte med oppdragsgiver angående bruk av biblioteket. Funksjonenene vi brukte for transformasjonen fant vi ved å lese gjennom online dokumentasjonen til OpenCV [28].

6.4.2 Prosess



Figur 20: Perspektiv transformasjon



Figur 21: Diagram over prosesseringssteg i perspektiv transformasjonen

Matrise konstruksjon (Figur 21 steg 1): Det blir laget to punktmatriser. En matrise inneholder de fire kilde-punktene, og en som blir konstruert slik vi ønsker at perspektivet skal være.

Om transformasjonen av bildet reduserer størrelsen til rammen fører det til tap av informasjon. For å miste minst mulig informasjon under perspektiv transformasjonen, så settes bredden og høyden til mål-kvadratet til kantene med størst lengde i kilde punktene (som vist på Figur 20).

Variablene under samsvarer med Figur 20 hvor L er avstanden mellom to rammer. Målpunktene ble satt opp slik:

$$P_0 = (L, L) \tag{6.1}$$

$$P_1 = (L + Width, L) \tag{6.2}$$

$$P_2 = (L + Width, L + Height) \tag{6.3}$$

$$P_3 = (L, L + Height) \tag{6.4}$$

GetPerspectiveTransform (Figur 21 steg 2):

Denne funksjonen fra OpenCV [28] lager en transformasjons-matrise som brukes i neste steg, ved bruk av kilde og målpunktene generert i forrige steg.

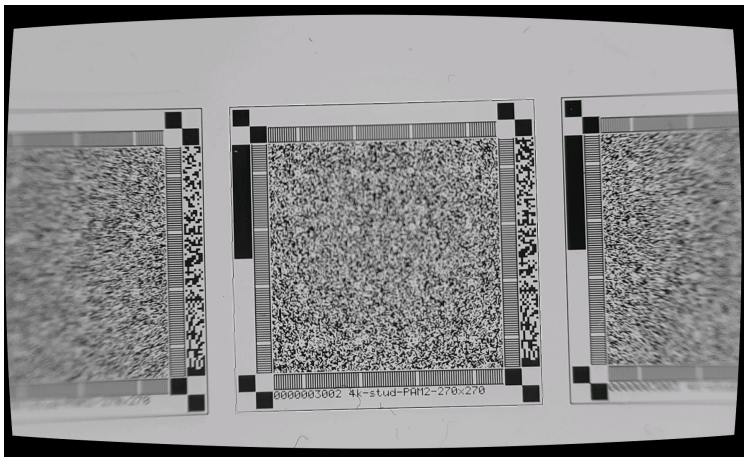
WarpPerspective (Figur 21 steg 3):

Denne funksjonen fra OpenCV [28] utfører selve perspektiv transformasjonen på bildet. Funksjonen tar også bredde og høyde slik at bildet blir transformert og croppet til:

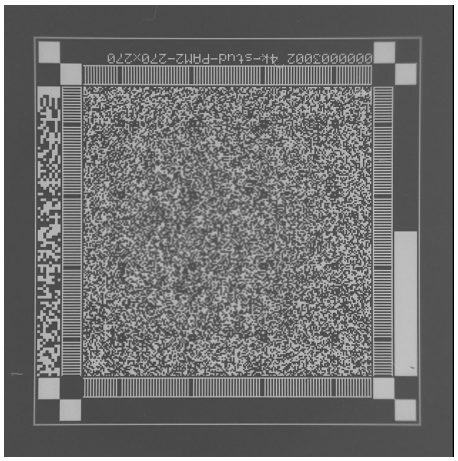
$$Size(x, y) = (2L + Width, 2L + Height) \tag{6.5}$$

Invertering (Figur 21 steg 4):

Til slutt inverteres bildet (kreves av Piql biblioteket). For å invertere bildet utføres en simpel operasjon. Hvert piksel i gråskala-bildet har en verdi på 0-255(forutsatt 8-bit per piksel), så alle piksler settes til 255 minus gråskala-verdien.



(a) Bilde av ramme med perspektiv.



(b) Bilde av beskjært, invertert og perspektiv-transformert ramme.

Figur 22: Før og etter bilder av perspektiv transformasjon.

6.5 Rotasjon

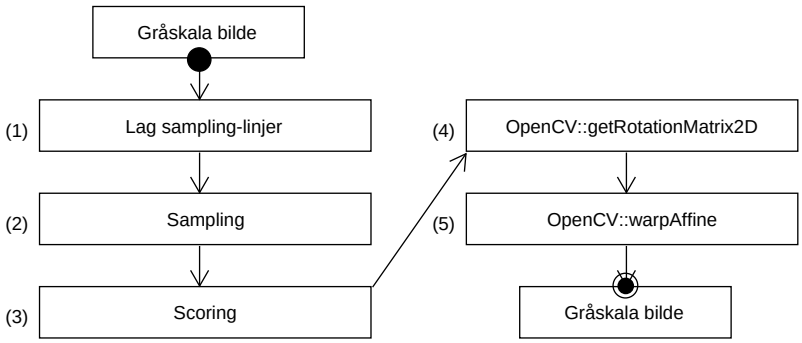
For at Piql biblioteket skal kunne lese data fra det perspektiv korrigerte bildet på Figur 22 (b), så må det være rotert slik at metadata området i rammen peker nedover i bildet som vist i Figur 1. Etter bildet er prosessert av denne modulen så er den klar til å bli prosessert av Piql biblioteket.

6.5.1 Fremgangsmåte

For å utføre rotasjonen lette vi først etter en metode for å regne ut hvilken vei rammen er rotert. Siden gråskala-verdi området er unikt i bildet og ligger langs en kant, så bestemte vi oss for å utføre rotasjon ut fra hvilken side av rammen området lå på.

Ved å gjenbruke og endre kode fra et feilet forsøk på markør deteksjon, hvor det ble konstruert en maske over bildet (se Figur 19a), så konstruerer rotasjons-modulen linjer hvor gråskala-området kan være. Siden halvparten av gråskala-området i ramme formatet er svart og den andre halvparten er være hvit, så laget vi en enkel scoring funksjon som regner ut forskjellen på gjennomsnittlig gråskala-verdi på begge halvdelene. For å utføre selve rotasjonen bestemte vi oss for å ta i bruk OpenCV for å spare tid.

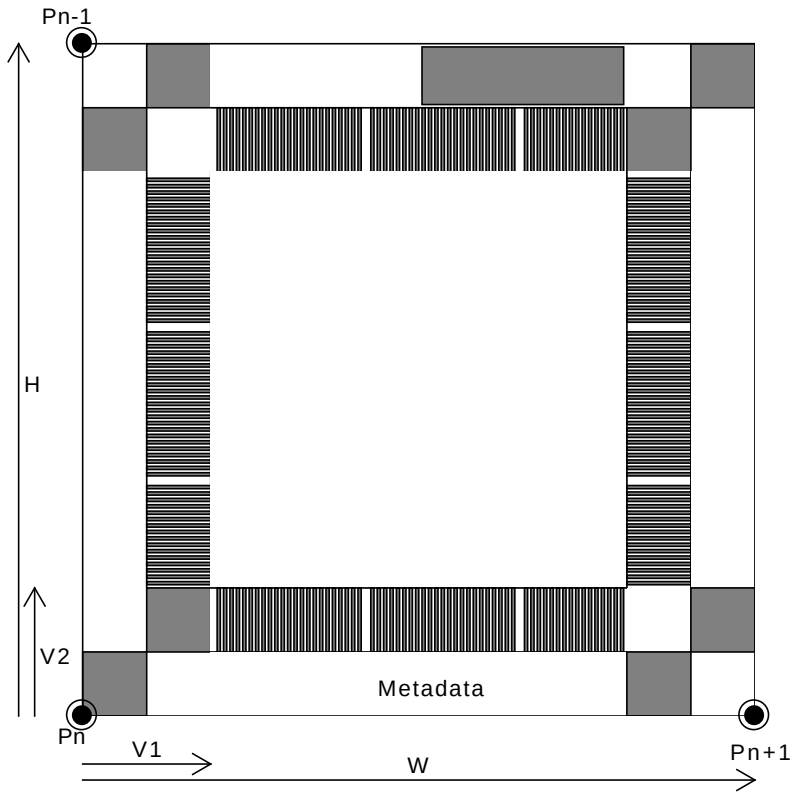
6.5.2 Prosess



Figur 23: Diagram over prosesseringssteg i rotasjons-modulen

For å finne ut hvilken vei rammen er rotert, scannes sidene på rammen etter området med gråskala verdiene (se nederst på Figur 1). Prosessen for å finne ut hvor mange grader bildet må roteres (0, 90, 180 eller 270 grader), beskrives i prosesseringssteg 1-3. Steg 4-5 beskriver hvordan selve rotasjonen blir utført.

Lag sampling-linjer (Figur 23 steg 1):

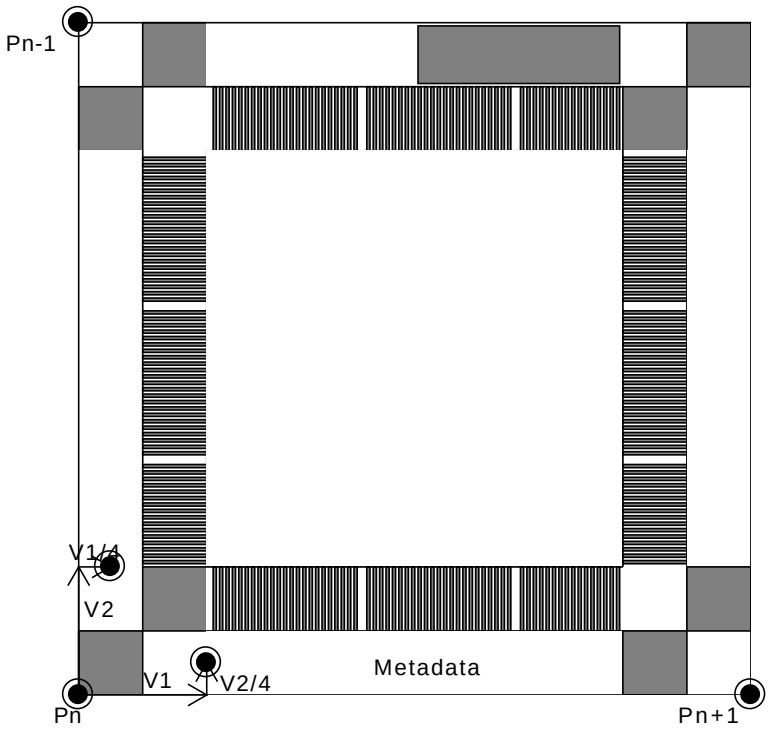


Figur 24: Diagram over konstruksjon av vektor V1 og V2

For hvert av hjørnepunktene konstrueres det to vektorer V1 og V2 med origo i hjørnepunktet, som vist på Figur 24. Lengden på kvadratet rundt en hjørne-markør er maksimum 15 % av avstanden mellom to nabo-punkter, derfor brukes 0,15 som en koeffisient for å beregne størrelsen på hjørne-masken. Vektorene er satt opp på følgende måte:

$$V1_n = 0,15 * (P_{n+1} - P_n) \tag{6.6}$$

$$V2_n = 0,15 * (P_{n-1} - P_n) \tag{6.7}$$



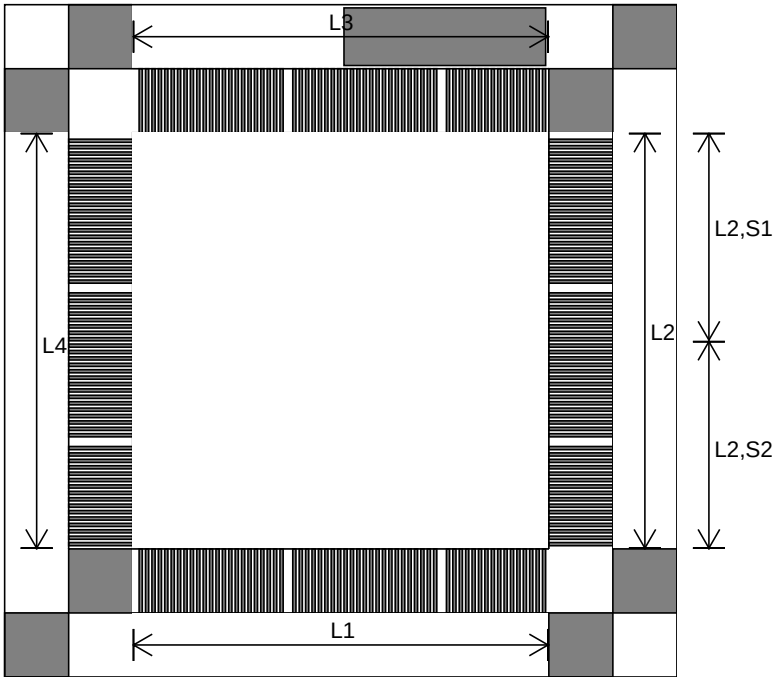
Figur 25: Diagram som viser konstruksjon av linjepunkter

Så konstrueres ende-punktene til sampling-linjene på følgende måte (som vist i Figur 25):

$$P2_{n-1} = V2_n + \frac{V1_n}{4} + P_n \tag{6.8}$$

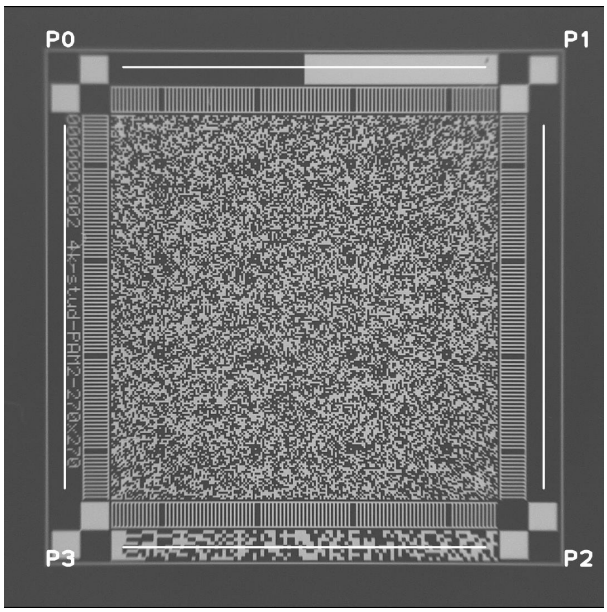
$$P1_n = V1_n + \frac{V2_n}{4} + P_n \tag{6.9}$$

P1 og P2 refererer til endepunktene av sampling linjene som trekkes parallelt med ytterkanten av rammen.



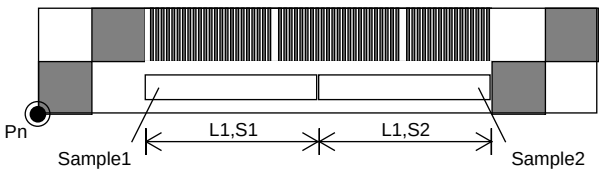
Figur 26: Diagram som viser konstruerte linjer for sampling

Ved å bruke de åtte punktene som ble funnet lages det fire linjer som igjen deles i to (som vist på Figur 26), som brukes i neste steg for sampling.



Figur 27: Rotert ramme med sampling-linje og punkt navn overlay

Sampling (Figur 23 steg 2):



Figur 28: Visualisering av sampling områder

Først blir det valgt ut åtte rektangulære områder i bildet på de åtte linjene funnet i forrige steg. Bredden (med y akse parallelt med linjen) til rektangelet settes til tre piksler. Så regnes det ut gjennomsnittlig gråskala verdi i de åtte rektanglene, som brukes i neste steg.

Scoring (Figur 23 steg 3):

Først regnes det ut forskjellen mellom gjennomsnittlig gråskala verdi funnet i forrige steg. Linjen med høyest score blir valgt som mest sannsynlig kandidat for å være gråskala-verdi området (som beskrevet på Figur 1). Nøyaktigheten til scoring-funksjonen er begrenset til rammer med to gråskala-verdier, ettersom forskjellen på hver halvdel av gråskala-verdi området er maksimal. Siden vi vet hvilken retning gråskala-verdi området ligger, så vet vi også hvor mange ganger bildet må roteres 90 grader.

GetRotationMatrix2D, warpAffine (Figur 23 steg 4-5):

GetRotationMatrix2D og warpAffine brukes til å rotere bildet 90, 180 eller 270 grader slik at metadata området peker nedover slik som i Figur 27.

6.6 Overlay generering

For å gi visuell tilbakemelding til brukere og utviklere under prosesseringen brukes overlay modulen. Denne modulen tegner linjer, tekst og rektangler over bildet før den vises på skjermen, slik at selve bildebehandlingen ikke blir påvirket.

Figur 29: Skjermdump av bilde med skanning område og FPS overlay

Modulen tar i mot informasjon for å tegne linjer, tekst og rektangler som vist i Figur 11. I tillegg utviklet vi en funksjon for å vise en tekst med FPS (bilder per sekund) som tegnes som overlay. Noen eksempler på overlay ligger i Figur 13, 16, 27 og 29.

7 Android applikasjonsutvikling

Ingen på gruppen hadde tidligere erfaring med å utvikle applikasjoner for smarttelefon. Selv om to på gruppen hadde en del kunnskap om Java var det fortsatt mye nytt å lære seg for å utvikle en Android applikasjon. Manifest, activities og layout var begreper vi ikke hadde hørt om før vi startet med oppgaven. Vi fant frem flere ressurser til å hjelpe oss med å lære mye rundt applikasjonsutvikling, bl.a. de offisielle ”Developer Guides” [34].

7.1 Oppsett av prosjektet

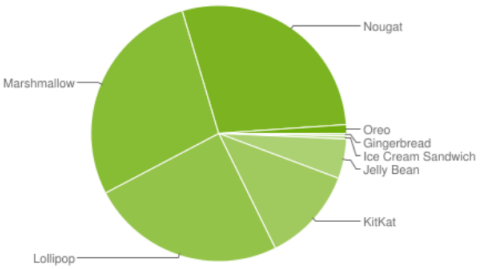
For å komme i gang med utviklingen var vi nødt til å sette opp et prosjekt i Android Studio hvor vi kunne utvikle applikasjonen. Android studio ble valgt til dette fordi:

- Den er en offisielt støttet IDE for utvikling av applikasjoner til Android.
- Det er et mye brukt verktøy som vedlikeholdes gjennom hyppige oppdateringer.
- Den inneholder flere verktøy laget spesifikt for utvikling og testing av Android applikasjoner.
- Den har en nyttig og omfattende dokumentasjon tilgjengelig online.

7.1.1 Valg av API nivå

Ved oppsett av et prosjekt i Android studio var vi nødt til å velge ut et minimum API nivå som enheten må støtte for å kunne kjøre applikasjonen. Applikasjonen skal være en demo applikasjon og skal ikke legges ut på Google Play Store, så det er ikke stort behov for å få med rundt 90% av brukere som Android anbefaler [35].

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%

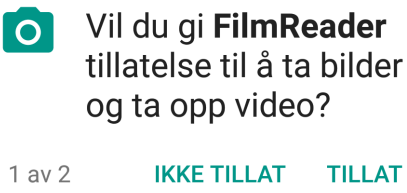


Figur 30: Android versjonsfordeling 5. Februar 2018

Selv om applikasjonen skal brukes som en demo ønsket vi å få med en del nyere smarttelefoner. Android 5.0 (API nivå 21) hadde store endringer i SDK hvor bl.a. den nye camera2 pakken kan komme til nytte, siden vi ønsker å hente ut høyoppløsnings bilder fra kamera. Android 5.0 og oppover hadde en ganske stor del av brukerene som vist på Figur 30, så vi valgte dette som minimum versjon for applikasjonen. For målversjon valgte vi 8.0 (API nivå 26) ettersom i en demonstrasjon er det gunstig å bruke den nyeste versjonen. Det var også anbefalt ifra Android å sette målversjon til den nyeste Android versjonen [35].

7.1.2 Tillatelser

For å kunne bruke kameraet i tillegg til å lagre filer på enheten krever Android brukerens samtykke. Brukeren blir spurt om tillatelse for å bruke kameraet til mobilen og for å lagre filer som vist på Figur 31. Svarer brukeren ”ikke tillat” på minst én av disse avsluttes applikasjonen ettersom den ikke vil fungere uten alle tillatelsene. Begge tillatelsene blir sjekket og eventuelt spurt om når applikasjonen starter dersom de ikke allerede er innvilget.



Figur 31: Brukergrensesnittet Android bruker for å spørre om tillatelser

7.2 Kamera

Etter vi fikk importert OpenCV biblioteket kom vi raskt i gang. Vi satte opp et JavaCameraView slik som vist i en eksempel fil som fulgte med OpenCV biblioteket. JavaCameraView gjorde det veldig lett å gjøre bildeoperasjoner på et bilde med funksjonen onCameraFrame som ble kalt for hvert bilde. Etterhvert i utviklingen viste det seg at JavaCameraView manglet en del funksjonaliteter som vi gjerne skulle hatt i vår applikasjon, spesielt høyere oppløsning.

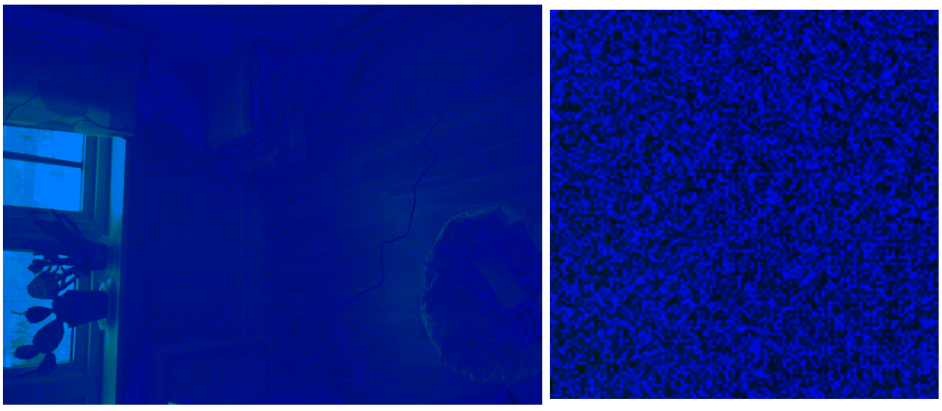
All håndtering av kamera skjedde i bakgrunnen og det var vanskelig å endre noe av dette ettersom vi da måtte inn i klassen for JavaCameraView inne i biblioteket til OpenCV og endre koden der. Å håndtere eventuelle feil ved oppstart av kameraet var det i tillegg lite støtte for. Vi startet å undersøke andre alternativer til å ta bilder og prosessere de i sanntid og endte opp med å se på kamera APIet til Android hvor det var to muligheter:

- Camera klassen, denne er foreldet i API nivå 21+ [36].
- ”android.hardware.camera2” pakken som støttes i API nivå 21+ [37]

Camera2 pakken er mye mer omfattende enn Camera klassen da man får mye mer kontroll rundt alt som har med kamera å gjøre på nyere smarttelefoner. Dette kommer på bekostning av mange klasser å forholde seg til derimot. Siden Camera klassen er foreldet i API nivå 21+ valgte vi å bruke camera2 pakken.

7.2.1 android.hardware.camera2

Camera2 pakken var vanskelig å ta i bruk. Det var mye oppsett rundt kameraet før man kunne begynne å ta bilder. Vi måtte lese mye i dokumentasjonen til camera2 for å skjønne hvordan pakken fungerte [37]. Vi forsøkte å velge ut de innstillingene som ga oss bilder med høyest kvalitet, ettersom vi ønsket å kunne lese ut data fra store rammer med tettpakket informasjon. Formatet på bilde satte vi til RAW_SENSOR, og oppløsningen satte vi til det høyeste som ble over 4K oppløsning på de nyeste smarttelefonene vi hadde. Etter et par hundre linjer med kode fikk vi endelig til å ta et bilde og lagre det til fil. For å lagre et bilde var vi nødt til å komprimere det til PNG. Komprimeringen skjer uten tap, men bildene ble fortsatt 15-18 MB store.



Figur 32: RAW_SENSOR bilde sammen med et nærbilde av det (vises best i farger).

Bilde ble det ikke helt som vi forventet. Det ble mørkt, blått og innehold mye støy

som vist på Figur 32. Bilde lignet veldig på blå-kanalen til en Bayer matrise [38]. Dette førte til at vi endret formatet og fikk ordentlige lyse og klare bilder ut som fortsatt hadde veldig god oppløsning. Det var ikke mange formater å velge mellom, men i endte opp på YUV_420_888 (også kjent som Y'UV420sp (NV21) [39]) som også er standard formatet for forhåndsvisning av kamera på Android [40]. Vi ønsket også å gjøre om bildene til kun gråskala ettersom ingen data blir tapt på dette i vårt tilfelle og det går raskere å prosessere i bildebehandlingen. YUV formatet har tre "plan" med data, Y, U og V. Y planet er kun gråskala så vi hentet ut Y-planet til bruk for prosesseringen.

Etter vi fikk satt opp camera2 APIet og tatt et bilde, fant vi ut at programflyten i koden var veldig dårlig. På grunn av dette gjorde vi om på mye på strukturen i programmet. Samtidig satte vi opp et rammeverk for å gjøre om et Image objekt [41] (output fra kamera) til et OpenCV matrise objekt [42] som vi sender videre til bildebehandlingsmoduler (se Kapittel 6).

7.2.2 Problemer

Å starte et kamera for å ta bilder var ikke problemfritt. Vi fikk ofte problemer med å starte applikasjonen på eldre smarttelefoner (mer i Kapittel 8). Vi klarte ikke å finne ut alle oppstartsfeilene både pga. manglende tid og kompetanse innenfor Android utvikling. I feilene som dukket opp var det vanskelig å finne ut hva som gikk galt. Senere i prosjektet fikk vi ikke til å reprodusere feilen og derfor heller ikke klart å fikse den.

En feilmelding vi ofte fikk var "CAMERA_IN_USE" som, i følge dokumentasjonen, betyr at kameraet som vi prøver å bruke allerede er i bruk [43]. Vi kom frem til at det er to mulige årsaker for feilmeldingen. Vi prøver å åpne kameraet to ganger, eller at en annen applikasjon bruker kameraet. Etter å ha undersøkt oppstarten av appen fant vi ut at koden vår kjøres dobbelt og appen prøver å åpne kameraet to ganger. Men etter å ha fikset dette oppsto fortsatt "CAMERA_IN_USE" feilmeldingen. Hypotesen om dette er at andre apper brukte kameraet avløste vi ganske fort ved å åpne flere andre apper som bruker kameraet.

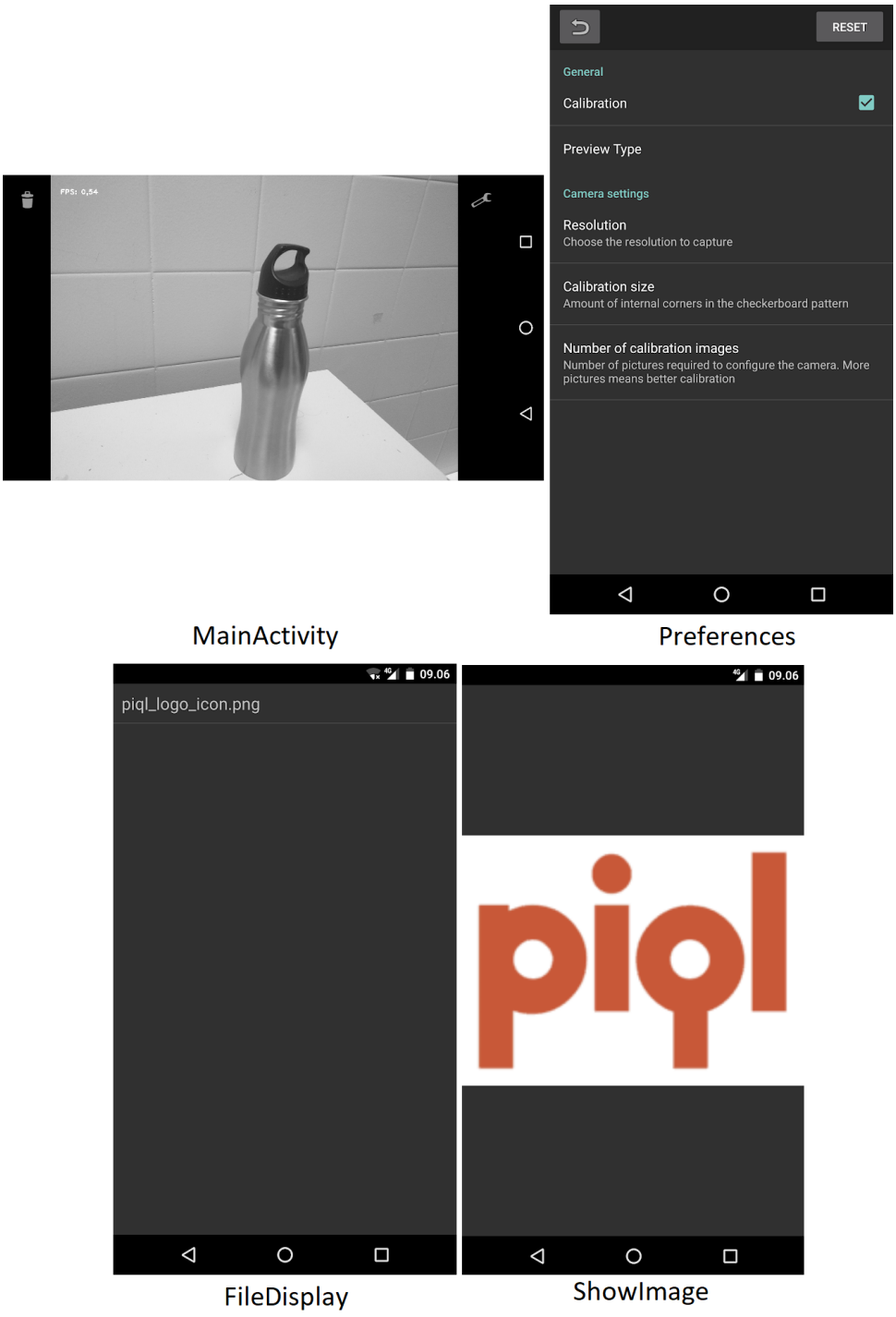
En annen feilmelding vi ofte fikk var "CAMERA_DISCONNECTED". Ifølge dokumentasjonen kan dette skyldes flere årsaker:

1. Kameraet er fysisk koblet ifra enheten.
2. Camera ID'en sendt med som parameter i openCamera metoden har blitt ugyldig.
3. Kamera tjenesten fjernet tilkoblingen fordi det kom en forespørsel til samme kamera med høyere prioritet.

Den første årsaken ble utelukket ettersom vi fint kunne bruke andre apper med samme kamera. Den andre og tredje årsaken var vi usikre på og hadde vanskeligheter med å finne ut hvordan vi skulle fikse.

Generelt var problemer rundt kamera ganske vanskelige å fikse ettersom feilmeldingene ga lite tilbakemelding og dokumentasjonen beskriver lite. Vi hadde heller ikke så mange enheter å teste feilen på.

7.3 Aktiviteter



Figur 33: 4 av de 5 aktivitetene som er i applikasjonen. Visning for tekstfiler mangler.

Aktivitetene satte vi opp slik at vi fikk skilt ut de forskjellige oppgavene til applikasjonen. Koden til aktivitetene blir også skilt ut i ulike Java filer (vises i klassediagrammet, Vedlegg C) så det blir en god struktur på koden under utviklingen.

7.3.1 MainActivity

MainActivity er hovedaktiviteten som applikasjonen kjører ved oppstart. Her åpnes kameraet med en gang og brukeren kan starte å kalibrere kameraet eller lese en dataramme. Alle andre aktiviteter initialiseres og kjøres herifra, og i tillegg blir alle bildebehandlings objektene initialisert. MainActivity består av tre views, ImageView og to knapper. Den ene knappen fører til Preferences aktiviteten, mens den andre knappen sletter konfigurasjonen. Hvis brukeren vellykket klarer å scanne en dataramme skifter appen aktivitet til FileDisplay. Vi forsøkte å bytte ut knappen for innstillinger med muligheten til å kunne sveipe aktiviteten inn ifra siden. Android Studio hadde funksjonalitet for å legge inn aktiviteter som var forhåndsutviklet, inkludert sveipe aktiviteter. Når vi prøvde dette ble det generert ny kode og nye filer. Dette ble fort komplisert så vi bestemte oss for å bruke enkle knapper. MainActivity er den eneste aktiviteten som er låst i landskapsmodus. Hvorfor blir forklart i Seksjon 7.3.4.

7.3.2 Preferences

Preferences, eller preferanser, er hvor brukeren kan endre innstillinger (Figur 33). Aktiviteten aksesseres ifra MainActivity og vi la til de innstillingene vi følte var nyttige, uten å bruke for mye tid på det. I tilegg la vi til et rammeverk slik at det enkelt kan legges til flere innstillinger i fremtiden. Vi la til følgende innstillinger i applikasjonen:

- Calibration: Om kameraet skal kalibreres eller ikke. Dette er en enkel checkbox som skrur kalibreringen på eller av. Se Seksjon 6.1 for mer informasjon om hva kalibrering er.
- Preview Type: Endrer på visningen til kameraet. Man kan velge mellom følgende typer:

  1. THRESHOLDED
  2. MARKERDETECT
  3. UNPROCESSED
  4. UNDISTORTED
  5. PROCESSED

  Alle disse er ikke implementert derimot. THRESHOLDED setter visningen av bilde til binært, se Seksjon 6.2.2. MARKERDETECT tegner område som brukes til å scanne etter en filmramme. Vi valgte å tegne denne uansett slik at brukeren får mindre problemer. De andre er ikke implementert utenom PROCESSED som har redundant kode.
- Resolution: Her kan brukeren skifte mellom å bruke forskjellige oppløsninger på kameraet. Lavere oppløsning gir bedre ytelse, men her begrenset vi brukeren fra å velge de aller laveste oppløsningene ettersom disse var ekstremt uklare og ga mange falsk-positive resultater. Alle oppløsninger som er mindre enn 700x700 i areal er ikke med i denne listen. Standard verdi på denne er høyest mulig oppløsning som på moderne smarttelefoner er ganske høyt.
- Calibration size: Her kan man stille på størrelsen på sjakkbrett mønsteret som brukes til å kalibrere med. Kalibrerings mønsteret vi fikk utdelt kom i tre størrelser. Den minste var for liten oppløsning, og dermed lite punkter å kalibrere etter. Den største hadde for mange punkter, slik at det ble vanskelig for kamera å finne punktene. Dermed gikk vi for det mellomstore kalibreringsmønsteret.

- Number of calibration images: Her velges det hvor mange bilder som skal bli tatt i kalibreringen. Høyere verdi her vil gi en bedre kalibrering, men også øke tiden det tar å kalibrere.

7.3.3 FileDisplay

FileDisplay aktiviteten kjøres når en dataramme er scannet. Ettersom Piql lagrer ".tar" filer på en dataramme, så la vi kun til funksjonalitet for å lese tar filer. Det var ingen støtte i Java for å lese tar filer så vi var nødt til å finne et tredjeparts bibliotek til å gjøre dette. Vi importerte Apache Commons Compress biblioteket som forklart i Seksjon 5.1. Filene hentes rekursivt fra ".tar" fila og vises i en og samme liste som dukker opp på skjermen. Vi valgte å ikke legge inn navigering av filer og mapper ettersom rammene er veldig begrenset i størrelse (4 KB) og vi antar at det ikke kommer til å være mange filer lagret på disse. Dermed valgte vi den enkleste måten som fikk jobben gjort, vise alle filene i en og samme liste.

Brukeren kan gå tilbake til MainActivity eller velge en fil. Vi la til støtte for visning av bilder (PNG og JPEG) og tekst filer i egne aktiviteter. For andre filtyper (f.eks. PDF) kan brukeren åpne den i en ekstern applikasjon installert på enheten.

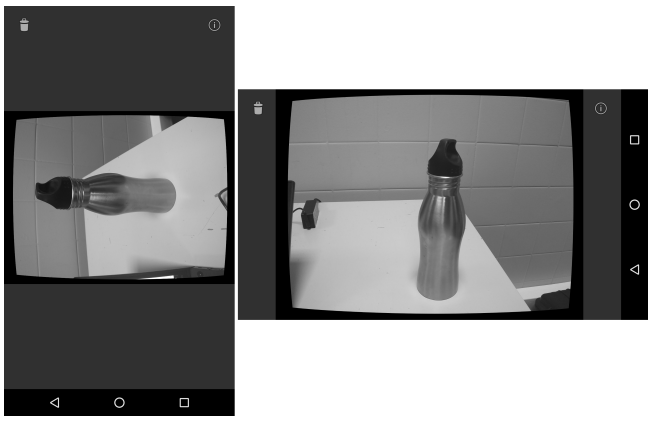
ShowImage og ShowText

Disse to aktivitetene inneholder bare en visning for å vise enten bilde eller tekst. Tekst visningen inneholder også funksjonalitet for å kunne bla nedover om teksten ikke passer på skjermen.

7.3.4 Problemer

Rotert kamera visning

I MainActivity fikk vi et problem med visningen av kamerabildene. Hvis aktiviteten var i portrett ble visningen bilde rotert 90 grader som vist i Figur 34. Samme hvilken vei enheten blir holdt så blir visningen av bilde rotert 90 grader mot klokka. Dette ble enda mer forvirrende når man skulle prøve å sikte seg inn på en ramme. Beveges enheten oppover går bildet på visningen til venstre.



Figur 34: Skjermdump av portrett- og landskapsmodus. Bilde i portrett blir både rotert og skalert ned.

Å rotere visningen var vanskeligere enn først antatt. Når vi forsøkte å roterte visningen ble den liten og begrenset i høyde. Som en midlertidig løsning låste vi aktiviteten til å kjøre i landskapsmodus. Senere fant vi en løsning ved å kombinere to ”tags” i XML filen til aktiviteten, ”rotation” og ”orientation”. Størrelsen på visningen ble fortsatt liten og dekket ikke hele skjermen i x- eller y-retning. Her gikk det ann å forstørre, men vi hadde ikke tilgang på de riktige verdiene til å regne ut den riktige forstørringen. Med det valgte vi å forkaste portrett modus og heller holde oss til landskaps modus i MainActivity.

Videresending av ”.tar” fil

Å sende ”.tar” filen da den kommer ut fra Piql sitt bibliotek, til FileDisplay aktiviteten viste seg å være ganske utfordrende. Vi måtte først sende ”.tar” filen tilbake til Java koden, også sende den over i en ny aktivitet. Android anbefaler å bruke ”Intent” klassen til å sende data gjennom aktiviteter. Skal egne klasser sendes over Intent er disse nødt til å gjøres ”Serializable” eller ”Parcelable”. Klassene som inneholder ”.tar” filen må da modifiseres. Disse klassene går under Apache lisensen som sier at man må dokumentere endringene. Istedet for å modifisere klasser vi ikke er kjent med, bestemte vi oss for å lagre hele filen lokalt på enheten og heller lese den inn igjen i FileDisplay. Filen lagres i selve applikasjonsmappen inne på enheten som man kun har tilgang til som administrator. Hver gang en ny ramme leses blir denne filen skrevet over. Om filen blir delt til tredjeparts applikasjoner så blir filen også lagret i en felles mappe på enheten.

Dårlig strukturert kode

Da vi skulle undersøke mange av feilene rundt kameraet og oppstart, fant vi ut at deler av koden kjøres dobbelt. I tillegg fant vi ut at applikasjonen kræsjer en del første gang den kjøres etter installasjon. Dette skyldes ufullstendig programmering fra oss, i tillegg til at noen funksjoner ikke fungerte som vi først trodde. Ettersom MainActivity skal tvinges i landskaps modus kjørte vi en funksjon som gjør dette når aktiviteten startes. Det vi ikke visste var at dette starter hele aktiviteten på nytt og dermed kjører mye kode dobbelt opp. Dette ble lett fikset ved å si i Manifest filen (fil som inneholder informasjon om applikasjonen) at aktiviteten skal kjøres i landskap modus. Ved å rydde opp i måten vi spør om tillatelser fikset vi alle kræsjene som dukker opp første gang man kjører applikasjonen.

7.4 Trådprogrammering

Etter vi kom i gang med bildebehandlingen la vi merke til at ytelsen fort ble verre. For å forsøke å løse dette problemet la vi til tråder slik at vi kunne prosessere flere bilder samtidig. Vi implementerte ikke dette pga. følgende grunner:

- Applikasjonen bruker allerede opptil 600MB minne (Seksjon 8.1.3), flere buffere for bilder ville gjort dette tallet enda høyere.
- Mye av koden i bildebehandlingen må skrives om for å legge til støtte for flere tråder. Vi hadde ikke nok tid til å fikse dette.

Vi la til én ekstra tråd for å skille bildebehandlingen slik at prosesseringen i bildebehandlingen ikke blokkerer for GUI tråden, som gjør at grensesnittet blir mer responsivt.

7.5 Internasjonalisering

Oppdragsgiver ønsket at applikasjonen skulle primært være på engelsk. Selv om det ikke var noe krav om internasjonalisering i applikasjonen, så la vi det til fordi det lett å legge til ved hjelp av Android Studio.

7.6 Implementasjon av brukergrensesnitt

Det endelige produktet fikk litt anderledes brukergrensesnitt i forhold til planene våres i Seksjon 4.2. Brukergrensesnittet vi endte opp med er vist på Figur 33. Under utviklingen fant vi ut at vi hadde bruk for to knapper, en for innstillinger og en for å slette konfigurasjonen. Vi valgte å fjerne informasjonsknappen for å ikke ha alt for mange knapper. Dekodingen av data gikk mye raskere enn forventet (under 50ms), så det var ikke nødvendig med en aktivitet for å vise at applikasjonen dekoder en ramme. Etter møte med Piql fant vi ut at det var behov for en aktivitet hvor man velger fil ettersom en ”.tar” fil kan inneholde flere filer.

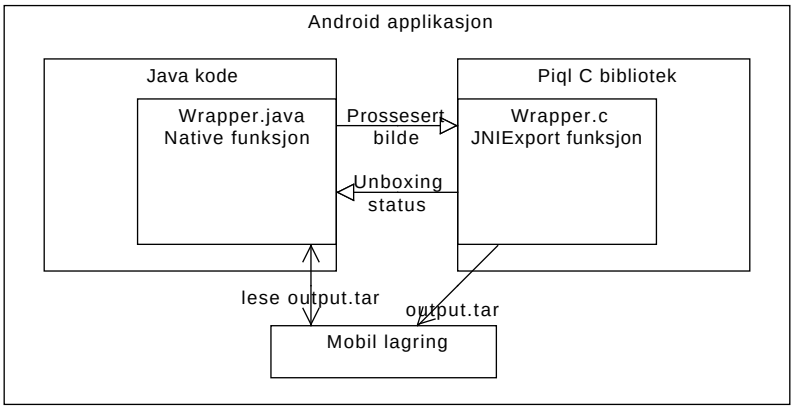
7.7 Piql bibliotek

Etter biblioteket var importert (se Kapittel 5.3) måtte det også implementeres i applikasjonen. For å bruke biblioteket i Java måtte vi lage et grensesnitt mot C og skrive kode i C som tar i bruk selve biblioteket. Vi hadde også noen problemer under utviklingen.

7.7.1 Java til C grensesnitt

For å kommunisere med Piql biblioteket som er skrevet i C lagde vi en egen klasse i Java som inneholdt en native funksjon. Denne klassen bruker JNI (Java native interface) for å kalle på en funksjon skrevet i C.

I ”wrapper.c” filen skrives en funksjon som tar i bruk Piql biblioteket, som synliggjøres for Java koden ved hjelp av JNIExport. Når native funksjonen i Java kalles (i wrapper.java), så kalles C funksjonen (i ”wrapper.c”) med samme parametre.



Figur 35: Oversikt over kommunikasjon over native C interface

Etter rotasjon er fullført (se Seksjon 6.5) kalles wrapper funksjonen med et parameter for høyde, bredde og det ferdig prosesserte bildet. Dersom data ble dekodet skrives det

en ”output.tar” fil til applikasjonens interne lagring og status returneres (som vist i Figur 35).

7.7.2 C wrapper

Kildekoden som ble skrevet for å ta i bruk koden internt i biblioteket befinner seg i ”wrapper.c” filen (se høyre side av Figur 35). Native funksjonen tar inn et bilde, og kaller på en funksjon i ”wrapper.c”, som passerer bildet sammen med to callback funksjoner som parameter videre til Piql biblioteket. Callback funksjonene sørger for å skrive resultatet til fil. Vi laget også en funksjon for logging slik at logg-meldinger fra biblioteket kan vises i Android studio.

7.7.3 Problemer

Under integrasjon av biblioteket hadde vi mange små feil fordi biblioteket ikke var laget for Android (se Kap 5.3). Det var også to mere tidkrevende og vanskelige feil:

- Applikasjonen krasjet tilfeldig under bruk av biblioteket på grunn av minne-korrupsjon (SIGBUS), som igjen oppsto på grunn av dobbel minne-frigjøring i biblioteket. Siden vi ikke hadde satt opp debuggeren i Android studio til å fungere med biblioteket var dette vanskelig å feilsøke. Etter rundt en uke feilsøking fant vi og rettet feilen, og det viste seg at feilen allerede var korrigert i den nyeste versjon av Piql biblioteket.
- Biblioteket krasjer når den prøver å aksessere feil minneadresse dersom et for lite bilde blir passert inn (SIGSEGV). På lik måte som forrige punkt var dette tidkrevende. For å unngå å passere inn bilder med for lav oppløsning begrenses oppløsningen til høyere enn 700x700.

7.8 Oversikt over implementasjon

For å vise oversikt over selve implementasjonen til applikasjonen har vi valgt å lage et klassediagram (Vedlegg C). Vi har her utelatt deler av implementasjonen som inneholder kode fra Piql biblioteket, på grunn av konfidensialitetsavtalen. Diagrammet bruker også vektor-grafikk, slik at den kan forstørres dersom rapporten leses digitalt.

8 Testing og kvalitetssikring

8.1 Testing

For å sikre oss at applikasjonen er stabil og fungerer slik som den skal, så har vi testet den på flere måter gjennom hele prosjektet.

8.1.1 Kontinuerlig testing

Under utviklingen har vi uformelt testet ny kode som vi utviklet. Ved å gjøre dette kontinuerlig fikk vi rettet opp i eventuelle feil som dukket opp underveis. Java har god støtte for automatiserte tester, men vi bestemte oss for å ikke ta i bruk disse ettersom testene er tidkrevende å skrive for bildebehandling.

Vi testet applikasjonen på våre egne smarttelefoner under utviklingen. Alle våre smarttelfoner har API nivå 21 eller høyere, så i utgangspunktet skal alle på gruppen kunne kjøre applikasjonen. Dette var ikke tilfellet med Samsung Galaxy S4 Active, hvor vi fikk problemer med å åpne kameraet. Etter å ha sjekket dokumentasjonen for feilen fikk vi ikke til å rette den opp og valgte å fortsette utviklingen. Da vi forsøkte å gjenskape feilen på et senere tidspunkt, fikk vi ikke til å reprodusere den.

8.1.2 Brukertesting

Mot slutten utførte vi en kort brukertest med oppdragsgiver. Testen ble utført via epost hvor vi sendte oppdragsgiver applikasjonen (Vedlegg I). Selv om brukertesten ikke var veldig omfattende, fikk vi korrigert to feil i applikasjonen som oppdragsgiver opplevde.

8.1.3 Endelig test

For å få en oversikt over ytelsen bestemte vi oss for å gjøre en større og mer omfattende og strukturert test etter utviklingsperioden. Vi opprettet flere tester som vi gjennomførte med flere smarttelefoner. Tre av de fem telefonene vi testet med kjører på forskjellige Android versjoner. Testene ga oss en oversikt over hvilken Android versjon applikasjonen fungerer best på. Siden smarttelefonen som ble brukt under utvikling hadde Android 7.1.1, var dette den mest stabile versjonen. Vi hadde en smarttelefon med Android 5.0.1 og en med 8.0 som vi også testet. Vi fikk noen problemer på disse, spesielt Android 5.0.1 hvor kameravisningen ble enten speilvendt eller strekt. Det er usikkert om årsaken til disse feilene var Android versjon eller maskinvaren siden vi kun hadde én mobil å teste Android 5.0.1 med.

Vi hadde planlagt å teste applikasjonen med flere forskjellige bakgrunnsbelysninger (styrke og farge). Dette var derimot ikke mulig med Slimlite LED som vi fikk ifra Piql (Seksjon 3.3). Fargelabben hadde en THOUSLITE LEDCube i rommet som vi fikk tilgang til (Seksjon 9.4.1) som hadde passet perfekt til å nøyaktig teste lesing av filmdata forskjellige bakgrunns-beslysninger. Vi kontaktet veilederen vår Sony for å få hjelp til å sette opp denne, men før vi fikk svar bestemte vi oss for å gjennomføre testene med Slimlite LED pga. tidsbegrensninger.

Figur 36: Oppsett for slutt tester

Oppsettet vårt for testingen ble som vist på Figur 36 hvor vi teipet film på lysbordet slik at filmen lå flat. Smarttelefonene var koblet til en PC underveis slik at vi kunne overvåke CPU bruk, minnebruk og se logg til eventuelle feil som måtte oppstå.

Utvalgte smarttelefoner

For testingen av applikasjonen forsøkte vi å bruke smarttelefoner med ulik hardware og Android versjon. Applikasjonen er begrenset til å kjøre på telefoner med API nivå 21 og høyere som tilsvarer Android 5.0. Vi spurte veilederene våres om NTNU hadde smarttelefoner de kunne låne ut til testing, men disse støttet ikke Android 5.0+. Vi brukte følgende smarttelefoner for testing:

- Samsung Galaxy A8 (2018)
- Samsung Galaxy S8
- Motorola Nexus 6
- Samsung Galaxy S4 Active

Galaxy A8 er samme mobil som vi fikk utdelt ifra Piql (Seksjon 3.3). Nexus 6 og S4 Active er våre egne smarttelefoner, og S8 fikk vi låne fra en medstudent.

Spesifikasjoner

Samsung Galaxy S4 active:

- Quad-core 1.9 GHz prosessor
- 2 GB RAM
- 16 GB intern minnekapasitet
- 8 MP kamera.
- Android versjon 5.0.1

Motorola Nexus 6:

- Quad-core 2.7 GHz
- 3 GB RAM
- 32/64 GB intern minnekapasitet
- 13 MP kamera
- 7.1.1

Samsung Galaxy A8:

- Octa-core (2x2.2 GHz & 6x1.6 GHz)
- 4 GB RAM
- 32 GB intern minnekapasitet
- 16 MP kamera
- 7.1.1

Samsung Galaxy S8:

- Octa-core (4x2.3 GHz & 4x1.7 GHz)
- 4 GB RAM
- 64 GB intern minnekapasitet
- 12 MP kamera
- 8.0.0

Tester

Vi ønsket først å se på om enheten klarer å kjøre, og ta i bruk applikasjonen. Vi målte også ytelsen under bruk av applikasjonen. Vi satte 1920x1080 (Full HD) som standard oppløsning på kameraet ettersom de fleste nyere mobilene støtter dette. Vi lagde følgende tester:

- Om applikasjonen starter opp.
- Om kalibreringen kan gjennomføres vellykket.
- Om kalibreringen lagres og lastes inn etter omstart.
- Om kalibreringen kan slettes.
- Om en dataramme gjenkjennes ved standard oppløsning.
- Om en dataramme dekodes og innholdet i filen kan vises.
- FPS ved minst mulig oppløsning.
- FPS ved standard oppløsning.
- FPS ved høyest mulig oppløsning.
- CPU bruk ved standard oppløsning.
- Minnebruk ved standard oppløsning.
- Minnebruk ved høyest mulig oppløsning.

CPU og minne måles ved hjelp av Android Profiler som er et innebygd verktøy i Android Studio. Denne måler CPU og minnebruk for applikasjonen uavhengig av ressursbruken til operativsystemet og andre applikasjoner. FPS målte vi ved bruk av overlay (Se Seksjon 6.6). Minste oppløsning er begrenset som beskrevet i Seksjon 7.3.2. Kalibreringen er sjeldent slått av under bruk, så vi bestemte oss derfor å ikke teste dette.

Resultater

	Galaxy A8	Nexus 6	S4 Active	Galaxy S8
Applikasjonen starter	Ja	Ja	Ja	Ja
Kalibrering gjennomføres	Ja	Ja	Ja	Ja
Laster inn kalibrering	Ja	Ja	Ja	Ja
Slette kalibrering	Ja	Ja	Ja	Ja
Gjenkjenne dataramme	Ja	Ja	Ja	Ja
Dekode dataramme	Ja	Ja	Nei	Ja
FPS minste oppløsning	8 (1024x768)	1,7 (1280x720)	1,6 (1280x720)	5 (1024x768)
FPS standard oppløsning	4	1,2	1 (1440x1080)	3
FPS høyste oppløsning	0,8 (4608x2592)	0,3 (4160x3120)	1 (1440x1080)	0,6 (4032x3024)
CPU standard oppløsning	ca. 15%	ca. 25%	ca. 30% (1440x1080)	ca. 20%
Minne standard oppløsning	130 - 170 MB	130 - 170 MB	50 - 60 MB (1440x1080)	130 - 170 MB
Minne høyeste oppløsning	500 - 600 MB (4608x2592)	300 - 400 MB (4160x3120)	50 - 60 MB (1440x1080)	400 - 450 MB (4032x3024)

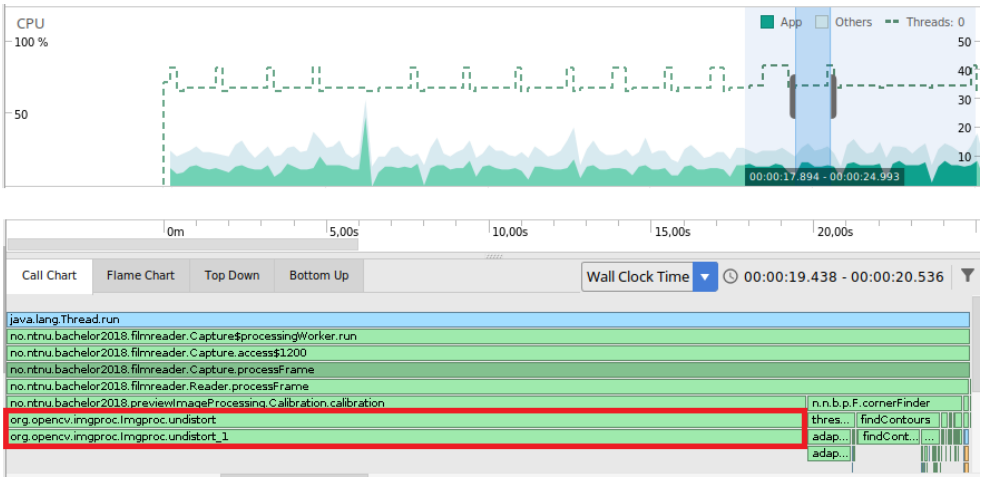
Tabell 1: Testresultater

Galaxy S4 Active støttet ikke den standard oppløsningen vi satte. Det nærmeste var 1440x1080 som også var den høyeste oppløsningen tilgjengelig. Grunnen til dette er mest sannsynlig at telefonen er fem år gammel. Dette er notert med parentes i Tabell 1.

Drøfting av resultater

Hovedfunksjonaliteten i applikasjonen fungerte som planlagt bortsett fra på S4 Active hvor vi ikke klarte å dekode en dataramme. Dette skyldes at bildet ble speilvendt i applikasjonen, som Piql sitt bibliotek ikke godtar. Vi fant ikke årsaken til speilingen.

Ytelsen i henhold til FPS er ganske dårlig. Det beste resultatet vi fikk var 8 FPS på Galaxy A8 sin minste oppløsning. Transformeringen kan ta opptil 80% av prosesseringen som vist nederst på Figur 37 (markert i rødt). Hvis kalibreringen er slått av vil FPS gå merkbart opp siden korrigering av forvrengninger ikke blir kjørt.



Figur 37: Logget CPU bruk under prosessering av et kamerabilde

CPU bruken er beregnet som % av CPU ut fra spesifikasjonene i Seksjon 8.1.3. Dette kan utnyttes bedre ved å ta i bruk flere tråder ettersom vi kun benytter én ekstra tråd. Øverst på Figur 37 ser vi at CPU bruken varierer. Den største CPU bruken var på S4 Active hvor bruken lå på rundt 30%. Bildebehandling tar mye CPU kraft. På en Galaxy A8 utnyttes kun 15% av en prosessorkraften, som ikke er optimalt. Om vi kunne økt denne prosenten til 30% eller 50% ved hjelp av flere tråder, så hadde FPS antageligvis nådd et mer akseptabelt nivå.

Minnebruken varierer veldig basert på hvilken oppløsning som er valgt. Det er flere bilder som holdes i minne under prosesseringen. Større oppløsning gir større bilder, som igjen fører til høyere minnebruk. Nexus 6 skilte seg ut ved å ha størst oppløsning, men lavere minnebruk enn Galaxy A8.

Når vi skulle teste Galaxy S8 oppdaget vi at ytelsen ble dårligere enn alle andre enheter, selv om den har bedre spesifikasjoner. CPU bruk og minnebruk målte vi med mobilen tilkoblet, så disse tallene kan være påvirket av dårlig FPS. Årsaken til dette er ukjent, men en midlertidig løsning var å ikke koble mobilen til datamaskinen.

8.2 Kvalitetssikring

8.2.1 Repository

Som nevnt i oppgavedefinisjon (Seksjon 1.2), så skal vi bruke et programvarebibliotek utviklet av Piql. De bruker SVN og CloudForge [44] for deres kode, og ønsket at vi også lastet opp vår kode på CloudForge i et repository som de opprettet. Da vi forsøkte å laste opp prosjektet vårt på CloudForge hadde vi en opplastningshastighet på 5Kbps som ikke er gunstig i vårt prosjekt. Vi kontaktet oppdragsgiver angående dette, men de visste ikke hvorfor vi hadde så lav opplastningshastighet. Vi hadde behov for et repository for prosjektet, så vi bestemte oss for å ta ibruk BitBucket som vi har brukt mye igjennom studiene her på NTNU. Vi spurte om og fikk lov av Piql til å laste opp kildekoden til applikasjonen på Bitbucket, som inneholder kildekoden til deres programvarebibliotek.

8.2.2 Dokumentasjon

For å kvalitetssikre koden da den skal overleveres til Piql i slutten av prosjektet, satte vi opp følgende prosedyrer:

- Alle funksjoner og klasser i applikasjonen skal dokumenteres slik at det JavaDoc kan generes.
- Alle variabler som er deklarert globalt i klasser skal kommenteres.
- Annen kode som har komplekst innhold skal kommenteres.

Vi har brukt ShareLaTeX, sammen med en LaTeX mal utviklet her på NTNU, for å skrive rapporten [45]. ShareLaTeX er basert på LaTeX som er et typesettingssystem for dokumentproduksjon [46]. Det gir oss muligheten til å redigere på samme dokument i sanntid, i tillegg til at alt lagres i skyen. Vi tok jevnlige sikkerhetskopier av alt vi hadde skrevet i ShareLaTeX.

9 Avslutning

9.1 Diskusjon

9.1.1 Resultat

Ved prosjektets slutt kom vi i mål og endte opp med en applikasjon omtrent slik som vi hadde forestilt oss. Applikasjonen fungerer ganske likt som en QR-kode applikasjon, bortsett fra at brukeren ”må” [1] kalibrere kameraet. Importering av Piql biblioteket fungerte til slutt, men tok lengre tid enn forventet.

Vi hadde ikke vært borte i Android applikasjonsutvikling i det hele tatt før prosjektet. Vi lærte mye om generell applikasjonsutvikling. Vi fant også ut at testing er viktig ettersom applikasjonen skal kjøre på mange forskjellige enheter med mye varierende maskinvare.

Bildebehandling

I starten av prosjektet hadde vi mye å lære om bildebehandling og applikasjonsutvikling. I planleggingsfasen anskaffet vi en bedre forståelse av bildebehandling. Vi leste om linse-forvrengninger, diskuterte løsningen med veiledere og Ivar Farup, leste bok om OpenCV [16] og testet OpenCV. Under utviklingen fikk vi også også erfaring med bildebehandlings-algoritmer og bruk av OpenCV.

Gjennom ulike uavhengige moduler prosesserer applikasjonen bilder slik at de kan dekodes av Piql sitt bibliotek. Det er mulige forbedringer til ytelse og nøyaktighet på søk og funksjonalitet. Deler av prosjektet kan også gjenbrukes til følgende:

- Kalibrere et kamera ved hjelp av OpenCV.
- Prosessere et bilde slik at det er klart for lesing av Piql sitt programvarebibliotek.
- Søke etter rektangulære sammenhengende kanter i et bilde.

Piql bibliotek

I begynnelsen da vi fikk tilgang til Piql biblioteket, så vi at de benyttet Qmake for å strukturere biblioteket. For å få et lignende resultat i Android Studio måtte vi sette oss inn i, og forstå hva make systemer er. Det var vanskelig å finne gode ressurser for å lære Cmake. Derfor tok vi i bruk CMakeConverter [24] for å sette opp et rammeverk for hvordan vi skulle bruke Cmake. Dette hjalp oss mye siden vi ikke visste hva vi måtte ha med i ”CMakeList.txt”.

Selv om vi hadde mange utfordringer og brukte lang tid, så klarte vi å ta i bruk Piql sitt bibliotek. Som nevnt i Seksjon 5.3 om importering av bibliotek, så er biblioteket skrevet for å scanne bilder med spesiallagde maskiner. Det vi oppnådde var å gjøre biblioteket tilgjengelig på Android, hvor biblioteket kan bli brukt i java kode.

[1] Hvis linse brukes bør man kalibrere, men det er aldri påkrevd

Android applikasjon

Brukeren har muligheten til å endre flere innstillinger for å tilpasse skanning av datarammer. Filvisningen forsøkte vi å gjøre enkel, ryddig og intuitiv slik at brukeren får en god oversikt over dataene som er lest ut. Problemene som sto igjen ved slutten av utviklingen var rotasjonen i portrett modus og oppstartsfeil. Oppstartsfeilene skyldes alltid kamera, og applikasjonen mangler funksjonalitet for å håndtere og gi tilbakemelding om dette. Som nevnt i Seksjon 7.2.2, klarte vi ikke å reprodusere disse feilene lenger. Oppdaging og dekoding av datarammer endte opp med å fungere relativt greit, mye på bekostning av FPS.

9.1.2 Alternativer / Valg underveis

QR-koder er det nærmeste vi kommer Piql sitt format, så vi forsøkte å gjøre litteratursøk på dette området tidlig i prosjektet uten suksess. Sony George hadde funnet et par artikler relatert til oppdaging av en QR-kode. Vi tok ikke nytte av disse artiklene siden metodene ofte benyttet seg av egenskaper QR formatet som **ikke** finnes i Piql sitt format (ulike hjørne-markører).

Utstyret som Piql ga oss var ikke **kritisk** for å komme i gang med prosjektet. Vi benyttet våre egne smarttelefoner imens vi ventet på smarttelefonen ifra Piql. Før vi fikk filmruller med datarammer, printet vi ut datarammer som Piql hadde sendt oss på A4 ark og skannet disse for å kunne teste applikasjonen. Datarammene vi printet ut på A4 ark gjorde vi store slik at vi ikke trengte å bruke linse for å skanne de.

Under planleggingen diskuterte vi om det var nødvendig med backend prosessering og dekoding. Dette valgte vi å ikke ta for oss fordi det krever at mobilen alltid er på nett og implementasjon av en oppegående server med høy sikkerhet. Dette ville gått mye utover implementasjon av kjernefunksjonaliteten til applikasjonen. Etter å ha testet OpenCV så vi tidlig i prosjektet at bildebehandling kunne utføres med grei hastighet i sanntid, så bruk av en ekstern server var unødvendig.

9.2 Kritikk av oppgaven

Oppgaven var veldig åpen, som viste seg å ha både fordeler og ulemper. En fordel var at vi bestemte mye av funksjonaliteten selv og hvordan vi ønsket å utføre arbeidet, men ulempen var at det ble mye prøving og feiling for å få et fungerende resultat. Vi kontaktet oppdragsgiver av og til for å spørre om problemer vi hadde med biblioteket og funksjonalitetsønsker, men det ble lite oppfølging av disse implementasjonene.

Vi tok ikke kontakt med oppdragsgiver tidlig nok etter vi fikk problemer med biblioteket deres. I tillegg var den første filmrullen vi fikk ikke helt riktig som forklart i seksjon 3.3. Feilen i Seksjon 7.7.3 om minnefrigjøring hadde Piql fikset i sitt repository, men da vi hentet biblioteket til bruk i vår applikasjon var ikke dette tilfellet. Vi brukte mye tid på å finne ut hva som forårsakte dette. Da vi endelig fant ut hva som var feilen viste det seg at Piql allerede hadde fikset dette. Vi sjekket aldri Piql sine oppdateringer på biblioteket så det er mulig vi gikk glipp av flere viktige endringer. Det var heller ikke bare å oppdatere biblioteket ettersom vi modifiserte en tidligere versjon i stor grad for bruk på Android.

9.3 Videreutvikling

Når det kommer til videreutvikling er det mye som kan forbedres. Den største modulen som kan forbedres er bildebehandlingen. Ytelsen i applikasjonen er allerede et problem som vist i Seksjon 8.1.3, og velger man større oppløsning på bildene blir det enda dårligere ytelse. Metodene og algoritmene som vi implementerte virket mest logisk for oss, men på grunn av vår begrensede kompetanse kan det hende det finnes bedre måter å oppdage en ramme med bedre ytelse. Implementasjon av flere tråder vil også kunne forbedre ytelsen ettersom vi kun kjører bildebehandlingen på én tråd (Seksjon 7.4).

Android Studio støtter kjøring av C og C++ kode. Vi implementerte bildebehandlingen i Java, men OpenCV er optimalisert i C og C++ så det kan være en fordel å se på mulighetene der. Det brukes også mange variable for å mellomlagre bilder under prosesseringen, bedre gjenbruk av buffere vil frie opp minnet mer.

Vi fikk ofte problemer med å kjøre applikasjonen på eldre smarttelefoner. Fiksing av oppstartsfeil (og eventuelt andre feil) vil styrke applikasjonen og gjøre den mer stabil og robust både på nyere og eldre smarttelefoner.

Applikasjonen mangler funksjonalitet for å kunne oppdage større oppløsninger på datarammer. Størrelsen er begrenset til 270x270 og dermed begrenses også mengden data som kan lagres på rammene til 4KB. Alle ”preview types” i preferences (Seksjon 7.3.2) er heller ikke implementert.

I etterkant fant vi ut at rotasjonsmodulen (Seksjon 6.5) kan også brukes til å speile hele bilde om filmen er lagt feil vei. Dette vil gjøre det lettere for brukeren siden det ikke er noen indikatorer på hvilken vei filmen skal ligge.

For å redusere feilfrekvensen til applikasjonen kan ”light radiance non uniformity” bli implementert. Selv om dette ikke er kritisk for at applikasjonen skal fungere, så vil det gjøre applikasjonen mer stabil.

9.4 Evaluering av gruppas arbeid

9.4.1 Organisering

Alle på gruppen ble enige om å møte opp klokken 8 om morgen for å kunne jobbe felles og strukturert med oppgaven. Dette gjorde vi i alle hverdagene slik at alle ble inkludert under hele prosjektet. Vi førte også timelogg hver dag, og gruppen endte opp med 1437 timer til sammen. Vi hadde ofte morgenmøter i fellesskap (stand-up meeting) hvor vi forsikret oss at alle hadde noe å gjøre. Kommunikasjon med oppdragsgiver gikk over Skype, e-post og Discord. De ukentlige møtene som vi hadde planlagt internt ble ikke utført. Vi tok disse møtene etter behov istedet.

Sony George kunne ikke norsk så vi måtte gjøre en vurdering på om vi skulle skrive rapporten på engelsk eller norsk. Vi valgte å skrive rapporten på norsk fordi vi har erfaring med å skrive på norsk fra tidligere emner. Marius Pedersen ble dermed veilederen vi henviste oss til for gjennomgang av rapporten. Kode og tekniske begreper var på engelsk, så det var vanskelig å finne ordentlige norske ord til rapporten.

For å få en naturlig rekkefølge i hoveddelen laget vi en avhengighetsgraf (se Vedlegg D). Det å lage avhengighetsgrafen ga oss en generell oversikt over hvordan hoveddelen i rapporten bør struktureres, slik at kapitler som påvirker eller inneholder teori som bør leses før et kapittel ligger ovenfor i rapporten.

Som nevnt i Seksjon 8.2.2 så brukte vi ShareLaTeX som platform for rapportskriving. Som verktøy bidro ShareLaTeX til godt samarbeid ettersom vi kan se på og gjennomgå det de andre på gruppen skriver i sanntid. Vi benyttet vurderingssystemet mye ved å markere tekst og gi det en merknad eller kommentar som påminnelser på at noe mangler her.

Når vi skulle skrive kritiske deler av applikasjonen eller satt fast i en problemstilling, ble parprogrammering brukt. Å få inn et nytt perspektiv på en problemstilling løste problemer effektivt og sparte oss mye tid. Vi brukte et Git repository for lagring av kode vha. Bitbucket, noe som fungerte bra siden vi allerede hadde erfaring med dette fra tidligere emner.

I starten avtalte vi med veilederene, Marius Pedersen og Sony George, at vi skulle ha ukentlige møter. Dette var en av anbefalingene vi fikk siden det er lettere å avlyse møter. Disse møtene holdt veilederene oppdatert på fremgangen slik at de kunne gi oss gode tilbakemeldinger basert på hvordan vi lå an. Marius Pedersen var veilederen vi spurte om alt administrativt rundt prosjektet og bacheloroppgaven. Om vi trengte noe ekstra spesifikt utstyr hadde Sony George styring på det meste av utstyret på fargelabben. For tekniske spørsmål til bildebehandlingen var begge veldig dyktige og foreslo flere fremgangsmåter. I tillegg fikk vi litt ekstra hjelp fra Ivar Farup tidlig i prosjektet.

Marius ga oss tilgang til et rom i kjelleren på fargelabben (A014) tidlig i semesteret. Dette har vært veldig nyttig for oss, da vi ikke måtte reservere rom eller lete etter sitteplass hver dag. Vi har også oppbevart alt utstyret vårt her isteden for å ta det med frem og tilbake fra campus.

Vi har inkludert Gantt skjema før og etter prosjektet som Vedlegg A og B. Enkelte moduler i bildebehandlingen ble kontinuerlig arbeidet med og forbedret gjennom hele utviklingsperioden, så disse milepælene ble satt til slutt.

Vi valgte å avslutte utviklingen 1 måned 1 uke før fristen for å få bedre tid til å skrive på rapporten. Etter hovedfunksjonaliteten var ferdig var det mye å skrive om ettersom vi ikke skrev mye underveis bortsett fra notater.

9.4.2 Fordeling av arbeidet

Håkon Heggholmen og Christian Hådem som kunne Java startet med utvikling av applikasjonen og bildebehandling. Java programmering var en ganske omfattende del av oppgaven. Når Even Måren Stende var ferdig med å importere Piql sitt bibliotek gikk han over til å skrive på rapporten og dokumentere hva vi hadde gjort istedet for å lære seg Java koden. Christian, som har hatt mattematikk 3, hadde et godt matematisk grunnlag til å gjøre bildebehandling, men valgte å ikke jobbe så mye med dette. Fordelingen av arbeidsoppgaver endte dermed opp som delvis prioritering av ferdigheter og hvilke oppgaver vi selv ønsket å jobbe med. Å importere Piql sitt bibliotek tok mye lengre tid enn vi trodde, så vi skulle satt flere personer på denne oppgaven i starten.

Selv om vi jobbet mye med kjente emner, var det mye nytt å lære seg. Even Måren Stende manglet erfaring med Java så fikk han ikke bidratt i store deler av utviklingen. Utenom dette utnyttet vi tiden godt med å jobbe mye parallelt med de forskjellige oppgavene.

9.4.3 Prosjektet som arbeidsform

Det å jobbe i en gruppe med utvikling har gitt oss mer realistisk erfaring med systemutvikling. Tidligere har vi funnet på eget prosjekt hvor vi bare skulle skrive rapporten, uten å utvikle det som skrives om. Vi har tidligere lært om teorien bak utviklingsmodeller, men gjennom prosjektet får vi viktig praktisk erfaring som kan brukes i arbeidslivet.

Oppgaven har opphav i virkelige problemer ulikt tidligere oppdiktede oppgaver fra arbeidskrav. Det at resultatet av prosjektet er av betydning til oppdragsgiver er også en ny og positiv erfaring. Oppgaven krever også at vi tar våre egne beslutninger og antagelser rundt krav, ønsker og mål til prosjektet.

I et prosjekt som var så åpent som denne var kommunikasjonen mellom oss og oppdragsgiveren viktig. I ettertid så vi verdien i å spørre oppdragsgiveren tidlig og ikke kaste bort tid om vi sitter fast. Det ble tatt flere antagelser under utviklingen ettersom oppdragsgiver ikke ønsket å fylle rollen som ”product owner”.

I starten ble vi anbefalt av veilederne til å skrive ned valg vi gjorde underveis i prosjektet. Vi fulgte anbefalingen og skrev ned hva vi hadde gjort underveis i utviklingen i et eget dokument sammen med hvorfor vi valgte å gjøre det slik. Dette gjorde det mye lettere når rapporten skulle skrives, spesielt når vi ikke jobbet mye med rapporten under utviklingen.

9.4.4 Oppfølging av Scrum

For utviklingsdelen av prosjektet hadde vi bestemt oss for å bruke Scrum som utviklingsmodell (Seksjon 1.6). Som verktøy benyttet vi oss av Trello tavler for utvikling og rapportskriving. På denne måten kunne alle på gruppen få en oppdatert oversikt på hva som var inkludert i gjeldende sprint og hva som var igjen i backloggen.

Hva fungerte bra

- Scrum ga oss et godt rammeverk for hvordan vi skulle sette opp utviklingen for å bli ferdig i tide.
- Sprinter ga oss muligheten til å sette krav til når enkelte deler av applikasjonen skulle være ferdig.
- Daily Scrum holdt utviklingen i gang og passet på at alle hadde noe å gjøre.

Hva fungerte dårlig

- Dokumentasjon av Scrum retrospekt og Scrum planning ble ikke godt utført Spesielt ikke i starten.
- Mye å forholde seg til, ikke alltid vi husket på å gjøre daily Scrum.
- Scrum har høyt fokus på innspill ifra product owner for å forsikre om at produktet blir slik kunden ønsker. Oppdragsgiver hadde lite krav til applikasjonen og ønsket ikke å være ”product owner” i vårt prosjekt.

Sprint oppdelingen ble brukt til å skille utviklingen av moduler. Uten en dedikert ”product owner” i prosjektet, ble deler av aktivitetene i Scrum ikke fulgt. Selv om vi ikke har brukt Scrum fullt ut så har det hjulpet oss med å få struktur på utviklingen og har vært et bra verktøy i prosjektet. Hvis oppdragsgiver hadde vært product owner måtte vi ha utviklet flere delversjoner av applikasjonen underveis. Vi ville derimot fått flere jevnlige brukertester sammen med tilbakemelding på hva Piql ønsker.

Kanban hadde muligens vært et bedre valg for utviklingsmodell siden den har høyt fokus på bruk av en tavle og visualisering av arbeidet. Modellen setter også en begrensning på hvor mange oppgaver som kan utføres samtidig. Vi hadde ikke satt oss et krav på hvor mange oppgaver som kunne utføres samtidig, men vi forsøkte å holde det så lavt som mulig til enhver tid.

9.5 Konklusjon

Oppgaven hadde en veldig unik og interessant bakgrunn, og samarbeidet i gruppen har gått greit ettersom vi kjenner hverandre ganske godt fra før.

Tiden som ble satt av i starten til å implementere de forskjellige modulene passet ganske bra med gjennomføringen. Vi fikk derimot ikke så god tid på slutten til å teste applikasjonen. Om vi hadde vært flinkere til å spørre veiledere og oppdragsgiver om problemer, hadde vi hatt mer tid til å utvikle de funksjonalitetene vi ikke rakk å implementere.

Slik applikasjonen er nå, kan det være risikabelt å kjøre en live demonstrasjon, spesielt på eldre enheter da vi selv fikk problemer med å åpne kamera på disse. Ytelsen er heller ikke god i applikasjonen (se Seksjon 8.1.3), mye pga. kalibreringen som er nødvendig når man bruker en makro-linse. Ser man bort ifra ytelsen kan applikasjonen brukes som et verktøy for å vise potensielle kunder teknologien bak digitalisering av data. Applikasjonen kan også brukes for å vise at digitaliseringen ikke er avhengig av eksisterende spesiallaget hardware (Piql Reader ™).

Bibliografi

[1] Papirbredden drammen kunnskapspark - piql as. Besøkt 12.01.2018. URL: http://www.papirbredden.no/cinevation.aspx.

[2] Piql as. generic 4k frame format - 4k frame structure description. Rev. 170630.

[3] Wikipedia contributors. 2018. Tar (computing) — Wikipedia, the free encyclopedia. Besøkt 19.04.2018. URL: https://en.wikipedia.org/w/index.php?title=Tar_(computing)&oldid=832525770.

[4] Commons, W. 2014. File:bayer pattern on sensor.svg — wikimedia commons, the free media repository. Besøkt 10.05.2018. URL: https://commons.wikimedia.org/w/index.php?title=File:Bayer_pattern_on_sensor.svg&oldid=139576558.

[5] Wikipedia contributors. 2018. Distortion (optics) — Wikipedia, the free encyclopedia. Besøkt 10.05.2018. URL: https://en.wikipedia.org/w/index.php?title=Distortion_(optics)&oldid=830036428.

[6] Commons, W. 2015. File:spherical aberration 2.svg — wikimedia commons, the free media repository. Besøkt 10.05.2018. URL: https://commons.wikimedia.org/w/index.php?title=File:Spherical_aberration_2.svg&oldid=155599123.

[7] Commons, W. 2013. File:chromatic aberration (comparison).jpg — wikimedia commons, the free media repository. Besøkt 10.05.2018. URL: https://commons.wikimedia.org/w/index.php?title=File:Chromatic_aberration_(comparison).jpg&oldid=107970907.

[8] Commons, W. 2018. File:saddle point.png — wikimedia commons, the free media repository. Besøkt 11.05.2018. URL: https://commons.wikimedia.org/w/index.php?title=File:Saddle_point.png&oldid=295931622.

[9] contributors, W. 2017. Opencv — wikipedia, the free encyclopedia. Besøkt 26.01.2018. URL: https://en.wikipedia.org/w/index.php?title=OpenCV&oldid=817334293.

[10] Activity | android developers. Besøkt 31.01.2018. URL: https://developer.android.com/reference/android/app/Activity.html.

[11] View | android developers. Besøkt 25.01.2018. URL: https://developer.android.com/reference/android/view/View.html.

[12] Design | android developers. Besøkt 10.04.2018. URL: https://developer.android.com/design/index.html.

[13] Google. Besøkt 18.01.2018. URL: https://www.google.no/.

[14] Group, S. M. Prisjakt - kunnskap for kjøp. Besøkt 18.01.2018. URL: https://www.prisjakt.no/.

[15] Gsmarena.com - mobile phone reviews, news, specifications and more... Besøkt 18.01.2018. URL: https://www.gsmarena.com/.

[16] Sommerville, I. 2011. *Software engineering 9th ed.* Addison-Wesley.

[17] Commons compress - overview. Besøkt 19.04.2018. URL: https://commons.apache.org/proper/commons-compress/.

[18] Apache license, version 2.0. Besøkt 08.05.2018. URL: https://www.apache.org/licenses/LICENSE-2.0.

[19] License - opencv library. Besøkt 08.05.2018. URL: https://opencv.org/license.html.

[20] Opencv releases download page. Besøkt 19.01.2018. URL: https://cmake.org/overview/.

[21] Qt 5.9. Besøkt 09.05.2018. URL: http://doc.qt.io/qt-5.9/index.html.

[22] Overview | cmake. Besøkt 09.05.2018. URL: https://cmake.org/overview/.

[23] Qt visual studio tools - visual studio marketplace. Besøkt 20.02.2018. URL: https://marketplace.visualstudio.com/items?itemName=TheQtCompany.QtVisualStudioTools-19123.

[24] Github - algorys/cmakeconverter. Besøkt 20.02.2018. URL: https://github.com/algorys/cmakeconverter.

[25] C standard library:string.h - clc-wiki. Besøkt 12.03.2018. URL: http://clc-wiki.net/wiki/C_standard_library:string.h.

[26] Camera calibration and 3d reconstruction. Besøkt 10.04.2018. URL: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[27] Feature detection. Besøkt 10.04.2018. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html.

[28] Geometric image transformations. Besøkt 12.04.2018. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html.

[29] Structural analysis and shape descriptors. Besøkt 10.04.2018. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html.

[30] Bradski, G. & Kaehler, A. 2008. *Learning OpenCV*. O'REILLY.

[31] Miscellaneous image transformations. Besøkt 12.04.2018. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html.

[32] Object detection. Besøkt 16.04.2018. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html.

[33] Bennett, S. & Lasenby, J. 2014. Chess – quick and robust detection of chess-board features. *Computer Vision and Image Understanding*, 118, 197 – 210. URL: http://www.sciencedirect.com/science/article/pii/S1077314213001999, doi:https://doi.org/10.1016/j.cviu.2013.10.008.

[34] Developer guides | android developers. Besøkt 12.03.2018. URL: https://docs.opencv.org/java/3.0.0/.

[35] Supporting different platform versions | android developers. Besøkt 12.04.2018. URL: https://developer.android.com/training/basics/supporting-devices/platforms.html#sdk-versions.

[36] Camera api | android developers. Besøkt 09.03.2018. URL: https://developer.android.com/guide/topics/media/camera.html.

[37] android.hardware.camera2 | android developers. Besøkt 09.03.2018. URL: https://developer.android.com/reference/android/hardware/camera2/package-summary.html.

[38] Wikipedia contributors. 2018. Bayer filter — Wikipedia, the free encyclopedia. Besøkt 10.05.2018. URL: https://en.wikipedia.org/w/index.php?title=Bayer_filter&oldid=834486695.

[39] contributors, W. 2018. Yuv — wikipedia, the free encyclopedia. Besøkt 09.03.2018. URL: https://en.wikipedia.org/w/index.php?title=YUV&oldid=824192271#Y%E2%80%B2UV420p_(and_Y%E2%80%B2V12_or_YV12)_to_RGB888_conversion.

[40] Imageformat | android developers. Besøkt 10.05.2018. URL: https://developer.android.com/reference/android/graphics/ImageFormat.

[41] Image | android developers. Besøkt 16.04.2018. URL: https://developer.android.com/reference/android/media/Image.html.

[42] Mat (opencv 2.4.2 java api). Besøkt 10.05.2018. URL: https://docs.opencv.org/java/2.4.2/org/opencv/core/Mat.html.

[43] Cameraaccessexception | android developers. Besøkt 25.04.2018. URL: https://developer.android.com/reference/android/hardware/camera2/CameraAccessException.html.

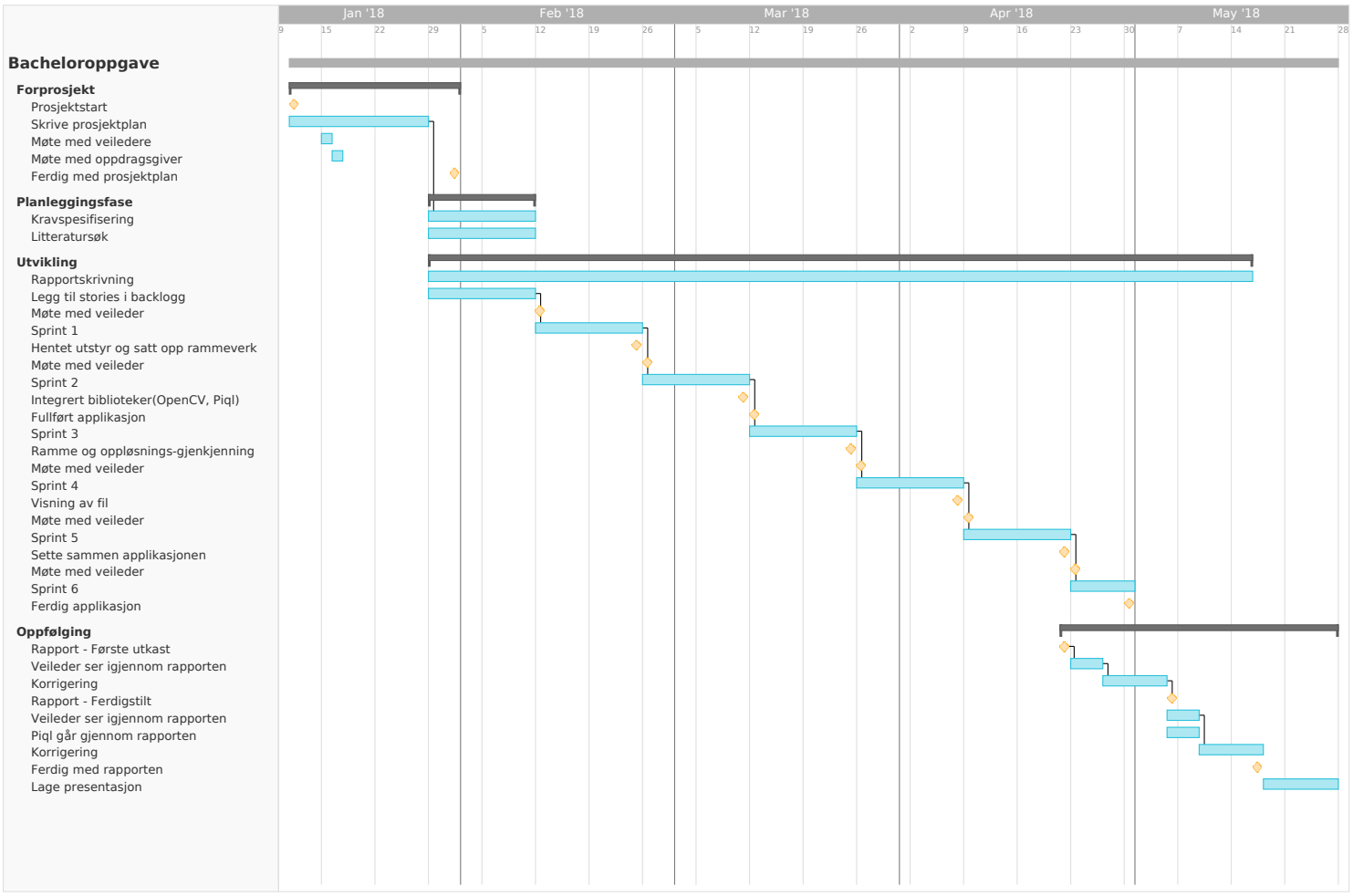
[44] Free subversion and git hosting | bug and issue tracking | cloudforge. Besøkt 09.05.2018. URL: http://www.cloudforge.com/.

[45] McCallum, S. & Farup, I. 2018. Bachelor thesis template (ntnu). Besøkt 09.03.2018. URL: https://github.com/COPCSE-NTNU/bachelor-thesis-NTNU.

[46] Wikipedia. 2018. Latex — wikipedia,. Besøkt 01.03.2018. URL: https://no.wikipedia.org/w/index.php?title=LaTeX&oldid=18299575.

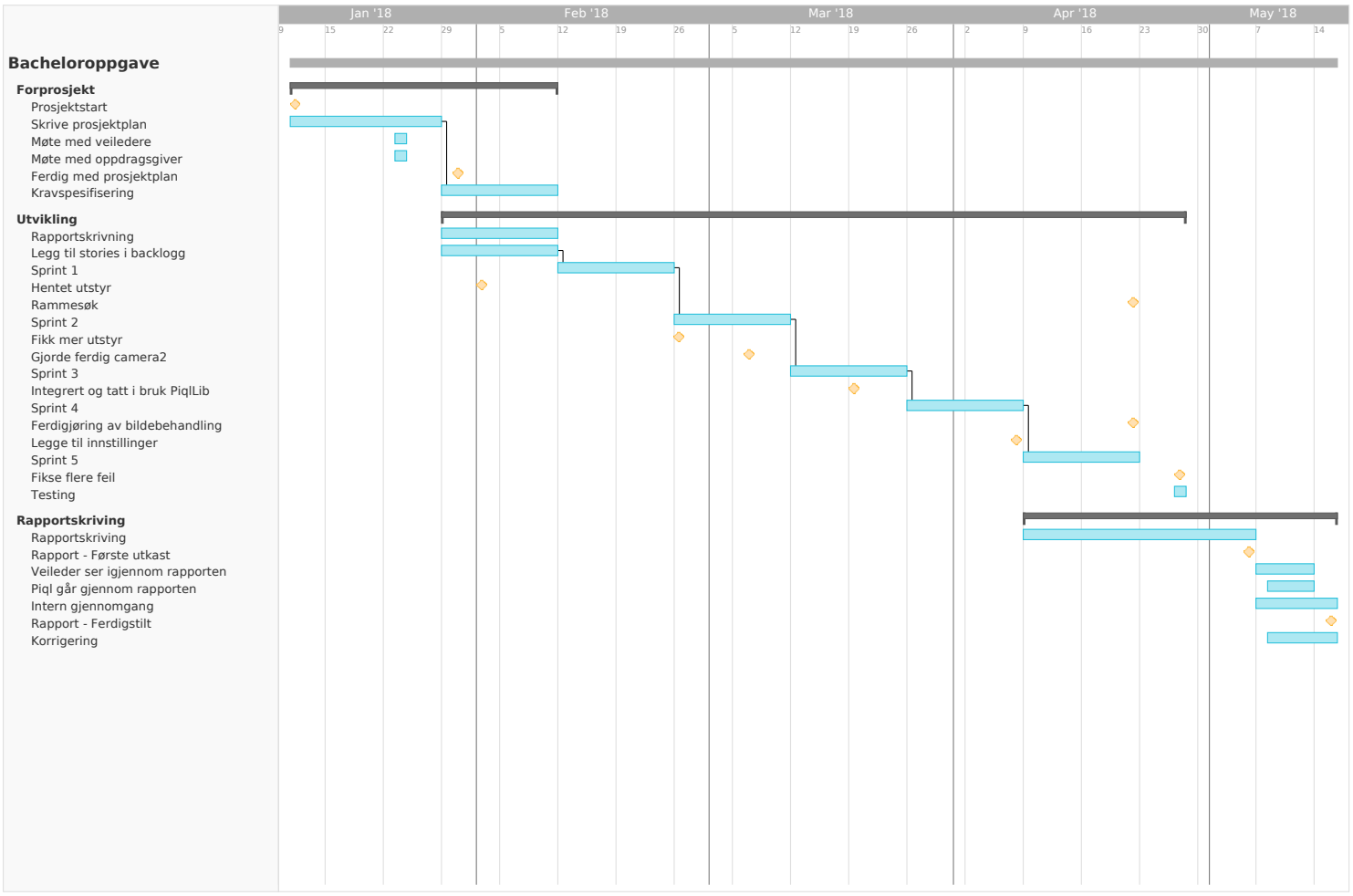
A Gantt, før

Gantt diagram av prosjektet laget under planleggingsfasen



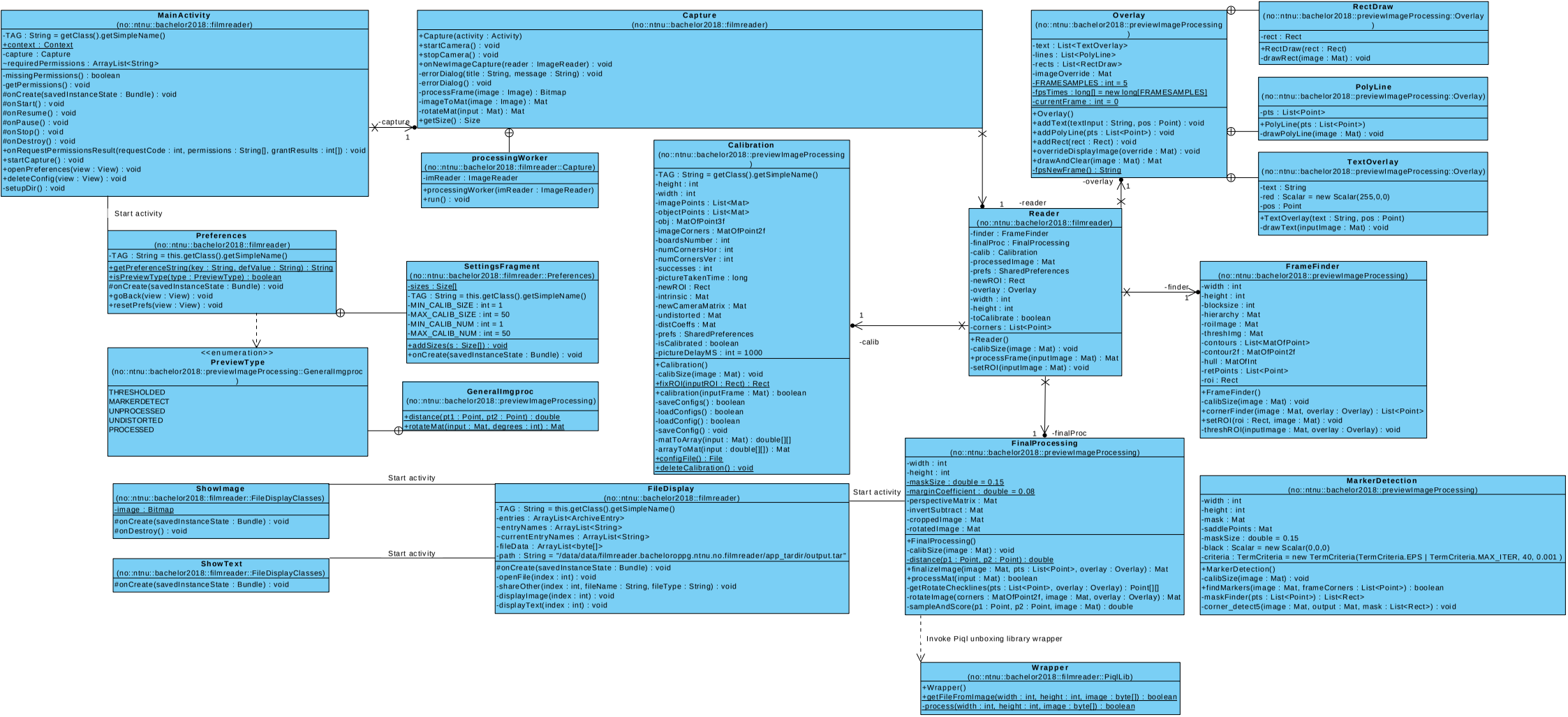
B Gantt, etter

Gantt diagram av prosjektet laget etter utviklingen var ferdig



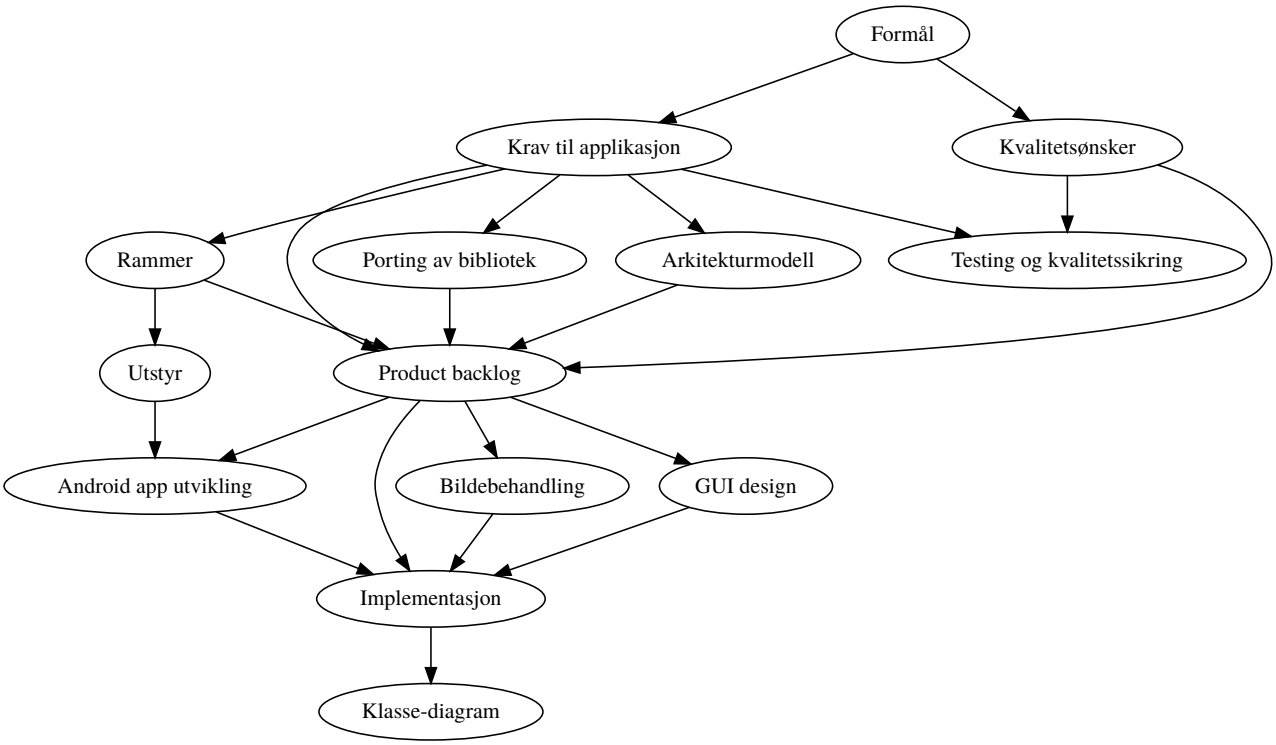
C Klassediagram

Diagram som viser hvordan klassene i applikasjonen ble implementert


MainActivity
(no::ntnu::bachelor2018::filmreader)
-TAG : String = getClass().getSimpleName()
+context : Context
-capture : Capture
~requiredPermissions : ArrayList<String>
-missingPermissions() : boolean
-getPermissions() : void
#onCreate(savedInstanceState : Bundle) : void
#onStart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
+onRequestPermissionsResult(requestCode : int, permissions : String[], grantResults : int[]) : void
+startCapture() : void
+openPreferences(view : View) : void
+deleteConfig(view : View) : void
-setupDir() : void
Capture
(no::ntnu::bachelor2018::filmreader)
+Capture(activity : Activity)
+startCamera() : void
+stopCamera() : void
+onNewImageCapture(reader : ImageReader) : void
-errorDialog(title : String, message : String) : void
-errorDialog() : void
-processFrame(image : Image) : Bitmap
-imageToMat(image : Image) : Mat
-rotateMat(input : Mat) : Mat
+getSize() : Size
-capture
1
processingWorker
(no::ntnu::bachelor2018::filmreader::Capture)
-imReader : ImageReader
+processingWorker(imReader : ImageReader)
+run() : void
Overlay
(no::ntnu::bachelor2018::previewImageProcessing
)
-text : List<TextOverlay>
-lines : List<PolyLine>
-rects : List<RectDraw>
-imageOverride : Mat
-FRAMESAMPLES : int = 5
-fpsTimes : long[] = new long[FRAMESAMPLES]
-currentFrame : int = 0
+Overlay()
+addText(textInput : String, pos : Point) : void
+addPolyLine(pts : List<Point>) : void
+addRect(rect : Rect) : void
+overrideDisplayImage(override : Mat) : void
+drawAndClear(image : Mat) : Mat
-fpsNewFrame() : String
RectDraw
(no::ntnu::bachelor2018::previewImageProcessing::Overlay
)
-rect : Rect
+RectDraw(rect : Rect)
-drawRect(image : Mat) : void
PolyLine
(no::ntnu::bachelor2018::previewImageProcessing::Overlay)
-pts : List<Point>
+PolyLine(pts : List<Point>)
-drawPolyLine(image : Mat) : void
TextOverlay
(no::ntnu::bachelor2018::previewImageProcessing::Overlay)
-text : String
-red : Scalar = new Scalar(255,0,0)
-pos : Point
+TextOverlay(text : String, pos : Point)
-drawText(inputImage : Mat) : void
-overlay
1
Start activity
Preferences
(no::ntnu::bachelor2018::filmreader)
-TAG : String = this.getClass().getSimpleName()
+getPreferenceString(key : String, defValue : String) : String
+isPreviewType(type : PreviewType) : boolean
#onCreate(savedInstanceState : Bundle) : void
+goBack(view : View) : void
+resetPrefs(view : View) : void
SettingsFragment
(no::ntnu::bachelor2018::filmreader::Preferences)
-sizes : Size[]
-TAG : String = this.getClass().getSimpleName()
-MIN_CALIB_SIZE : int = 1
-MAX_CALIB_SIZE : int = 50
-MIN_CALIB_NUM : int = 1
-MAX_CALIB_NUM : int = 50
+addSizes(s : Size[]) : void
+onCreate(savedInstanceState : Bundle) : void
<<enumeration>>
PreviewType
(no::ntnu::bachelor2018::previewImageProcessing::GeneralImgproc
)
THRESHOLDED
MARKERDETECT
UNPROCESSED
UNDISTORTED
PROCESSED
GeneralImgproc
(no::ntnu::bachelor2018::previewImageProcessing)
+distance(pt1 : Point, pt2 : Point) : double
+rotateMat(input : Mat, degrees : int) : Mat
Calibration
(no::ntnu::bachelor2018::previewImageProcessing
)
-TAG : String = getClass().getSimpleName()
-height : int
-width : int
-imagePoints : List<Mat>
-objectPoints : List<Mat>
-obj : MatOfPoint3f
-imageCorners : MatOfPoint2f
-boardsNumber : int
-numCornersHor : int
-numCornersVer : int
-successes : int
-pictureTakenTime : long
-newROI : Rect
-intrinsic : Mat
-newCameraMatrix : Mat
-undistorted : Mat
-distCoeffs : Mat
-prefs : SharedPreferences
-isCalibrated : boolean
-pictureDelayMS : int = 1000
+Calibration()
-calibSize(image : Mat) : void
+fixROI(inputROI : Rect) : Rect
+calibration(inputFrame : Mat) : boolean
-saveConfigs() : boolean
-loadConfigs() : boolean
-loadConfig() : boolean
-saveConfig() : void
-matToArray(input : Mat) : double[][]
-arrayToMat(input : double[][]) : Mat
+configFile() : File
+deleteCalibration() : void
1
-calib
Reader
(no::ntnu::bachelor2018::filmreader)
-finder : FrameFinder
-finalProc : FinalProcessing
-calib : Calibration
-processedImage : Mat
-prefs : SharedPreferences
-newROI : Rect
-overlay : Overlay
-width : int
-height : int
-toCalibrate : boolean
-corners : List<Point>
+Reader()
-calibSize(image : Mat) : void
+processFrame(inputImage : Mat) : Mat
-setROI(inputImage : Mat) : void
1
-reader
-finder
1
FrameFinder
(no::ntnu::bachelor2018::previewImageProcessing)
-width : int
-height : int
-blocksize : int
-hierarchy : Mat
-roiImage : Mat
-threshImg : Mat
-contours : List<MatOfPoint>
-contour2f : MatOfPoint2f
-hull : MatOfInt
-retPoints : List<Point>
-roi : Rect
+FrameFinder()
-calibSize(image : Mat) : void
+cornerFinder(image : Mat, overlay : Overlay) : List<Point>
+setROI(roi : Rect, image : Mat) : void
-threshROI(inputImage : Mat, overlay : Overlay) : void
1
-finalProc
FinalProcessing
(no::ntnu::bachelor2018::previewImageProcessing)
-width : int
-height : int
-maskSize : double = 0.15
-marginCoefficient : double = 0.08
-perspectiveMatrix : Mat
-invertSubtract : Mat
-croppedImage : Mat
-rotatedImage : Mat
+FinalProcessing()
-calibSize(image : Mat) : void
-distance(p1 : Point, p2 : Point) : double
+finalizeImage(image : Mat, pts : List<Point>, overlay : Overlay) : Mat
+processMat(input : Mat) : boolean
-getRotateChecklines(pts : List<Point>, overlay : Overlay) : Point[][]
-rotateImage(corners : MatOfPoint2f, image : Mat, overlay : Overlay) : Mat
-sampleAndScore(p1 : Point, p2 : Point, image : Mat) : double
Invoke Piql unboxing library wrapper
Wrapper
(no::ntnu::bachelor2018::filmreader::PiqlLib)
+Wrapper()
+getFileFromImage(width : int, height : int, image : byte[]) : boolean
-process(width : int, height : int, image : byte[]) : boolean
MarkerDetection
(no::ntnu::bachelor2018::previewImageProcessing)
-width : int
-height : int
-mask : Mat
-saddlePoints : Mat
-maskSize : double = 0.15
-black : Scalar = new Scalar(0,0,0)
-criteria : TermCriteria = new TermCriteria(TermCriteria.EPS | TermCriteria.MAX_ITER, 40, 0.001)
+MarkerDetection()
-calibSize(image : Mat) : void
+findMarkers(image : Mat, frameCorners : List<Point>) : boolean
-maskFinder(pts : List<Point>) : List<Rect>
-corner_detect5(image : Mat, output : Mat, mask : List<Rect>) : void
Start activity
ShowImage
(no::ntnu::bachelor2018::filmreader::FileDisplayClasses)
-image : Bitmap
#onCreate(savedInstanceState : Bundle) : void
#onDestroy() : void
Start activity
ShowText
(no::ntnu::bachelor2018::filmreader::FileDisplayClasses)
#onCreate(savedInstanceState : Bundle) : void
FileDisplay
(no::ntnu::bachelor2018::filmreader)
-TAG : String = this.getClass().getSimpleName()
-entries : ArrayList<ArchiveEntry>
~entryNames : ArrayList<String>
~currentEntryNames : ArrayList<String>
-fileData : ArrayList<byte[]>
-path : String = "/data/data/filmreader.bacheloroppg.ntnu.no.filmreader/app_tardir/output.tar"
#onCreate(savedInstanceState : Bundle) : void
-openFile(index : int) : void
-shareOther(index : int, fileName : String, fileType : String) : void
-displayImage(index : int) : void
-displayText(index : int) : void
Start activity

D Avhengighetsgraf

Graf som viser hvordan vi strukturerte hoveddelen i rapporten


Formål
Krav til applikasjon
Kvalitetsønsker
Rammer
Porting av bibliotek
Arkitekturmodell
Testing og kvalitetssikring
Utstyr
Product backlog
Android app utvikling
Bildebehandling
GUI design
Implementasjon
Klasse-diagram

E Prosjektplan

Prosjektplan - Bacheloroppgave

Håkon Heggholmen Even Måren Stende
Christian Hådem

Vår 2018

Innhold

1 Mål og Rammer

1.1 Bakgrunn

Piql AS er et norsk selskap med over 30 kontorer rundt om i verden som utvikler teknologi og løsninger innen langtidslagring av digitale data. Selskapets kjernekompetanse er innen overføring av digitale filer fra og til fotosensitiv film [1]. Kundene til Piql er foreløpig store institusjoner som f.eks. arkiver og biblioteker som har viktig informasjon og historie som de ønsker å ta backup av.

For å oppnå langtidslagring/arkivering av data, så har Piql tatt i bruk fotosensitiv film. Med et QR-kode lignende proprietært format som Piql har utviklet, så kan data printes binært på filmen og leses av ved hjelp av spesialisert hardware (Piql writer og Piql reader). Piql ønsker nå å se på muligheter ved å lese data med smarttelefoner, for å demonstrere til potensielle kunder robustheten ved deres eksisterende lagringstjeneste.

Oppgaven er dermed å lage en mobil applikasjon for å lese data fra film og presentere den.

Oppgaven fant vi da oppdragsgivere kom å presenterte oppgavene sine til studentene her på NTNU i Gjøvik. Piql sin oppgave skilte seg ut og vekket oppmerksomhet i gruppen vår.

Om oss
Vi er tre fulltidsstudenter som studerer dataingeniør på NTNU i Gjøvik. Vi har gått i klasse sammen i over to år og er kjent med mye av det samme stoffet. Når det gjelder programmeringsspråk har vi erfaring med C, C++, SQL, Python og enkel PHP og x86 Assembly. To av oss har også hatt en del Java og applikasjonsutvikling mens én har hatt programvaresikkerhet. Andre relevante kunnskaper som vi har er algoritmiske metoder og systemutvikling, som kommer godt i bruk i store prosjekter.

1.2 Prosjektmål

Målet med oppgaven er å utvikle en mobil applikasjon som kan lese filer fra eksisterende film. Gjennom prosjektet må vi undersøke metoder og prinsip-

per innenfor bildebehandling, Android utvikling og kamerateknologi. Bredden i prosjektet er vanskelig å estimere, slik at hovedmålet er å utvikle selve applikasjonen, hvor vi selv står fritt til å gjøre forbedringer.

Resultatmål
Lage en mobil applikasjon som ansatte hos Piql kan bruke til å:

- Demonstrere uavhengighet av eksisterende spesialisert hardware for fremtidig uthenting av data.
- Vise hvordan man kan i prinsippet hente ut data ved bruk av et kamera og en prosessor.
- Hente ut og vise filer fra en filmrute/dataramme som bruker Piql sitt proprietære format.

Effektmål

- Øke salg av langtidslagringstjenester hos Piql.
- Bygge tillit til eksisterende lagringstjenester på film.

1.3 Rammer

Ethvert prosjekt har begrensede ressurser og rammer det skal holdes innenfor. Vi har dermed laget en oversikt over de viktigste begrensningene og rammene som vi er nødt til å holde oss innenfor under hele prosjektet.

Tids og økonomiske rammer
I henhold med prosjektet vil vi få utdelt utstyr for å teste produktet. Piql har gitt oss et budsjett på 8000kr som skal gå til en telefon og en linse som vi selv velger ut. Dette vil ikke koste gruppen noe økonomisk, vi får derimot ansvaret med å levere tilbake utstyret i samme tilstand som det ble mottatt.

Vi skal skrive en rapport av prosjektet, denne har innleveringsfrist 16.05.2018. Denne vil gradvis bli jobbet med gjennom hele prosjektet, men vi er nødt til å sette av en god del av tiden mot slutten til å skrive ferdig og finpusse denne etter som dette er hele vurderingsgrunnlaget vårt.

Juridiske rammer
Vi har signert en konfidensialitetsavtale med Piql og er dermed nødt til å følge Piql sine retningslinjer når det gjelder konfidensiell informasjon. Det

er Piql som får eierrett på kildekoden som blir laget for applikasjonen, så vi har ikke lov til å ta med koden videre til andre selskaper.

Tekniske rammer
Ved valg av operativsystem står vi fritt til å velge mellom utvikling på Android eller iOS. Vi har valgt å gå for kun Android siden alle på gruppen inkludert oppdragsgiver bruker Android telefon til daglig, og Android gir oss mye mer frihet under og etter utvikling. Etter ønske ifra Piql skal appliksjonen primært være på engelsk og all kode og kodekommentarer skal være på engelsk.

2 Omfang

2.1 Fagområde

Piql har et ønske om å utvikle en mobil applikasjon. Dermed vil fagområdet primært være mobil applikasjonsutvikling. I seksjon 1.3 har vi sagt at vi skal utvikle applikasjonen for Android, mobil applikasjonsutvikling på Android foregår for det meste i Java. Bibloteket ifra Piql som vi skal ta i bruk er i C, så vi vil antageligvis såvidt komme borti dette og. Applikasjonen skal og ta seg av bildebehandling som vi antar vil bli en stor del av prosjektet. Noen viktige deltemaer her blir objektgjenkjenning og transformering.

2.2 Avgrensning

Som utledet i tekniske rammer i seksjon 1.3, skal applikasjonen begrenses til Android utvikling. Bildebehandlingen vil for det meste bestå av å analysere, transformere og lese ut data ifra et bilde. Det vil ikke være noe fokus på sikkerhet i applikasjonen ettersom det kun vil brukes som en demo. Applikasjonen vil heller ikke ha noe lagring av data eller kommunikasjon med andre enheter.

2.3 Oppgavebeskrivelse

Piql ønsker primært en mobil applikasjon for å lese data ifra film. Med dagens mobilkamera krever dette ekstra utstyr, bl.a. en makro-linse for å ta nærbilde av dataområdet på filmen.

Applikasjonen skal fungere omtrent akkurat som en QR-scanner hvor brukeren tar et bilde med mobiltelefonen, applikasjonen gjenkjenner koden, leser ut data og presenterer det for brukeren enten det måtte være tekst, et bilde eller noe annet.

Applikasjonen skal dermed åpnes rett i kameraet hvor brukeren kan ta et bilde av film med data. Bilde må bli tatt gjennom en makro-linse, som nevt ovenfor, ettersom dataen er har høy oppløsning på et lite område. Bildet av datarammen fra kameraet må behandles slik at den ligner mest mulig på kilde bildet før den ble printet ut. Om dette gjøres på frontend, backend eller begge deler er opptil oss å argumentere for, og finne ut av hva som er best. Applikasjonen skal kun presentere dataen, ikke lagre den, men dette må gjøres forskjellig uavhengig om det er tekst, bilde, pdf eller noe annet.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Å ha en prosjektleder var et av rådene vi fikk med denne oppgaven. En prosjektleder er en person som tar store beslutninger i prosjektet og passer på at ting går etter planen. Ved intern diskusjon har vi kommet frem til at prosjektlederen vil være Håkon Heggholmen. Han står også for kommunikasjon mellom Håkon G. Larsson som er prosjektets oppdragsgiver.

Christian er dokumentasjonsansvarlig og skal passe på at gruppen fører timelogg, og dokumenterer beslutninger/endringer de har gjort i forhold til hva som var planlagt. Ellers er det ikke behov for mer spesifikk inndeling av rollene.

Eventuelt utstyr som blir lånt ut til gruppen under prosjektet er Even sitt ansvar å levere tilbake slik som det kom. Vi har også undertegnet en konfidensialitetsavtale som beskrevet i de juridiske rammene i seksjon 1.3.

3.2 Rutiner og regler i gruppa

3.2.1 Rutiner

Det er beregnet at hver person skal jobbe 30 timer i uken med prosjektet. For å garantere dette har vi satt opp en timeplan for prosjektarbeid. Gruppen skal møte opp på avtalte steder minst 4 dager i uken (mandag, tirsdag, torsdag, fredag), med mindre annet er diskutert internt. Gruppen skal også ha ukentlige møter hvor det diskuteres hva som er gjort og hva som må gjøres. Disse møtene skal det skrives korte referat ut av og en Trello tavle skal oppdateres med eventuelle nye oppgaver.

3.2.2 Grupperegler

For å få et godt samarbeid på gruppen med god effektivitet har vi satt opp grupperegler. Disse skal hjelpe til med å avgjøre interne problemer som måtte oppstå under prosjektets gang.

Fravær
Alle gruppemedlemmer skal møtes opp til avtalt tid, fravær må meldes ifra til gruppen på forhånd. Ved lengre fravær (over en uke kombinert umeldt/ubegrunnet fravær) skal veileder eller eventuelt andre bacheloroppgave ansvarlige kontaktes for å diskutere situasjonen. Hvis vedkommede fortsatt ikke har kommet med en gyldig begrunnelse på det punktet, vil han/hun bli fjernet ifra gruppen.

Konflikter
Ved uenigheter av implementasjon og mindre beslutninger rundt prosjektet skal hele gruppen tas med for å diskutere uenigheten. Dette gjelder kun beslutninger som ikke har innvirkning på prosjektet som helhet, f.eks. grafiske design-valg og andre reversible valg.

Ved uenigheter av større beslutninger/vedtak i prosjektet skal et møte settes opp internt i gruppen hvor alle partene får tid til å komme med sine egne argumenter. Disse møtene blir dokumentert sammen med motargumenter. Løser ikke dette konflikten blir veileder kontaktet.

Andre regler
Alle parter skal konsekvent gi konstruktive tilbakemeldinger til hverandre og

passe på at alle alltid har noe å gjøre i prosjektet. Skulle noen bli distrahert over lengre tid må de settes i fokus på prosjektet. Om man skulle sitte fast i en oppgave skal man prøve å få hjelp fra andre i gruppen. Sitter alle fast, prøv å ta kontakt med veileder eller andre grupper.

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Applikasjonen vil bestå av flere deler. Den første delen blir å lage en fungerende applikasjon, som henter ut bilder fra kameraet. Den andre delen av prosjektet er selve bildebehandlingen, her må bilde prosesseres og applikasjonen er nødt til å få data ut av bilde. Den tredje delen er å presentere data enten det måtte være tekst, bilde, pdf etc.

Bildebehandlings delen er den delen med mest usikkerhet rundt, ettersom vi ikke har erfaring med å lese binær data ifra bilder. Vi har veiledere med erfaring innenfor området som kan hjelpe oss med dette, men vi antar at funksjonaliteten bak bildebehandlingen/prosesseringen vil ta mye tid å utvikle.

Før rapporten skal innleveres den 16.mai.2018 må prosjektet være ferdig, derfor blir det mindre fokus på utviklingen i mai.

4.1.1 Noen karakteritistika ved vårt prosjekt

- Under utviklingen vil det være mye avhengighet mellom funksjonalitet i deler av programmet. Dette medfører at mange moduler i programmet må fullføres i en bestemt rekkefølge.
- Det er lite krav om oppfølgning og verifisering under utvikling fra Piql sin side. Dette kan erstattes med verifisering fra veiledere.
- På grunn av rapporten som skal skrives, så må beslutninger under utvikling dokumenteres og begrunnes i rapporten.

4.1.2 Argumentasjon for valg av modell

Plandrevne utviklingsmodeller inneholder mye dokumentasjon og planlegging og er best egnet for større prosjekter. Smidige utviklingsmodeller derimot er bedre egnet for mindre prosjekter og prosjekter hvor kunden vil følge med under utviklingen og passe på at produktet blir som det skal. Selv om det fort kan høres ut som om smidige passer best i vårt tilfelle, har vi studert begge sider for å finne ut hvilken metode som passer best.

I vårt prosjekt er det en relativt liten applikasjon som skal utvikles, dermed er ikke **Fossefall** veldig aktuell. Ettersom det oppstår endringer under utviklingen, så vil det ikke være mulig å lage en fullstendig plan og kravspesifikasjon. **RUP** har mange av de samme punktene som i fossefall, så dette vil heller ikke være gunstig.

Inkrementell vil fungere bra siden produktet består av flere deler. Ulempen er at vi må inn å estimere hvor lang tid det kommer til å ta å utvikle hver del. Det baserer seg også mye på testing og tilbakemelding ifra kunden, men Piql hadde ingen ønsker om delleveranser. Delvis utvikling av prosjektet blir mer eller mindre inkludert i de fleste smidige utviklingsmodeller.

Smidige utviklingsmodeller har mange gode punkter for vårt prosjekt. Enkle og små oppgaver ifra kunden i **XP** er veldig relevant i vårt prosjekt. Ettersom vi forventer å måtte gjøre endringer under utvikling som ikke stemmer med kravspesifikasjonen, så vil refactoring være viktig. Parprogrammering kan være nyttig på enkelte viktige moduler i applikasjonen.

Fordelen med **Scrum** er at vi kan ha en backlog med oppgaver. Backloggen kan være nyttig ettersom det kan dukke opp flere ønsker til funksjonalitet underveis. Med tanke på rapporten som vi skal skrive kan det være veldig nyttig å ha mål som vi kan gå for, men ikke nødvendigvis må bli ferdig med.

I en såpass uerfaren gruppe som oss kan **Kanban** komme godt til nytte. Kanban tar i bruk en enkel tavle hvor arbeidet blir delt opp og organisert. Med dette blir det mindre "switching" rundt på arbeidsoppgaver og man får god oversikt over hva som trengs å gjøres.

Med disse argumentene har vi ved diskusjon kommet frem til at vi velger **Scrum**, men vi vil legge til mer krav til dokumentasjon underveis i prosjektet.

- Vi er en liten gruppe på tre personer.

- Ved bruk av backlog kan vi lettere balansere skriving av rapport og sette opp riktig rekkefølge på implementasjon av funksjonaliteter.
- Scrum gir bedre samarbeid med oppdragsgiver/product owner(Piql), som får tilgang til nyeste versjon av programvaren.

En sprint vil være 2 uker lang og vi vil ha en daily scrum hver dag i perioden. Vi valgte 2 uker fordi dette er det mest vanlige. Én uke blir litt for kort siden det ikke er noe krav til oppfølging under utviklingen. 3 uker er derimot litt for langt for å sette oss realistiske og fornuftige mål under hver sprint. "Product owner" blir Håkon G. Larsson og siden "ScrumMaster" fungerer omtrent som prosjektleder, så tar Håkon Heggholmen denne rollen.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Hver mandag morgen så skal det være et statusmøte i gruppen, hvor vi finner ut hvordan vi ligger an i henhold til planen. Vi har planlagt å ha møter med veileder en gang i uken under planleggingsfasen, og heller ta møte hver andre uke under utviklingen. Hvis dette er mer enn nødvendig enkelte uker vil møtene bli avlyst på forhånd.

5 Organisering av kvalitetessikring

For å sikre kvalitet og konsistens i prosjektet har vi opprettet interne regler og standarder som skal følges. Vi tar og i bruk flere verktøy for å hjelpe oss med analysering og utføring av prosjektet.

- **ShareLaTeX** gir oss felles tilgang til dokumenter hvor vi kan skrive sammen, opprette issues, få god oversikt over filer, i tillegg til å lage pdf dokumenter med bra utforming.
- **Google sheets** for enkel og rask timelogging.
- **Android Developer Studio** for utvikling av Android applikasjoner.
- **TeamGantt** hadde gratis tilgang for et prosjekt opptil tre brukere. Her var det lett å opprette å modifisere gantt skjemaer.

Mer i seksjon 5.2.

5.1 Dokumentasjon, standardbruk og kildekode

Vi står ganske fritt når det gjelder standardbruk i kildekoden. Koden og kodekommentarene skal skrives på engelsk og skal være i et repository som vi får tilgang til av Piql. Et dokument med kodestandarder skal opprettes internt, her skal det stå regler om struktur, innrykk, dokumentasjon og kommentering, osv. om koden.

Det skal ukentlig dokumenteres hva som er gjort en uke av valg, og argumentasjon av valg, slik at dette kan refereres tilbake til når den endelige rapporten skal skrives. All kildekode skal dokumenteres godt slik at ingen forvirring eller andre problemer oppstår i etterkant.

5.2 Konfigurasjonsstyring

Vi vil få utdelt brukere med tilgang til et repository hvor vi skal utvikle applikasjonen. Piql vil følge med her under utviklingen. Disse brukerkontoene vil også ha lesetilgang til Piql sitt repository, som inneholder kildekode for avlesning av data fra ferdig prosessert bilde. TortoiseSVN vil bli brukt som versjonskontrollværktøy. Bugs skal rapporteres som issues i repositoriet hvor kritiske bugs skal prioriteres. Er det blokkerende bugs som står i veien for funksjonaliteten av applikasjonen må de fikses med en gang.

5.3 Risikoanalyse

Før prosjektet starter er det viktig å analysere risikoene knyttet til prosjektet. Dermed kan vi analysere hvilke tiltak som burde gjøres for å redusere konsekvensene assosiert med prosjektets integritet.

Konsekvens/ Sannsynlighet	Liten	Middels	Høy	Kritisk	Katastrofal
Lite sannsynlig					5
Noe sannsynlig		4		1, 6	
sannsynlig			3		
Meget sannsynlig				2	
Svært sannsynlig					

Tabell 1: Gradering av risiko

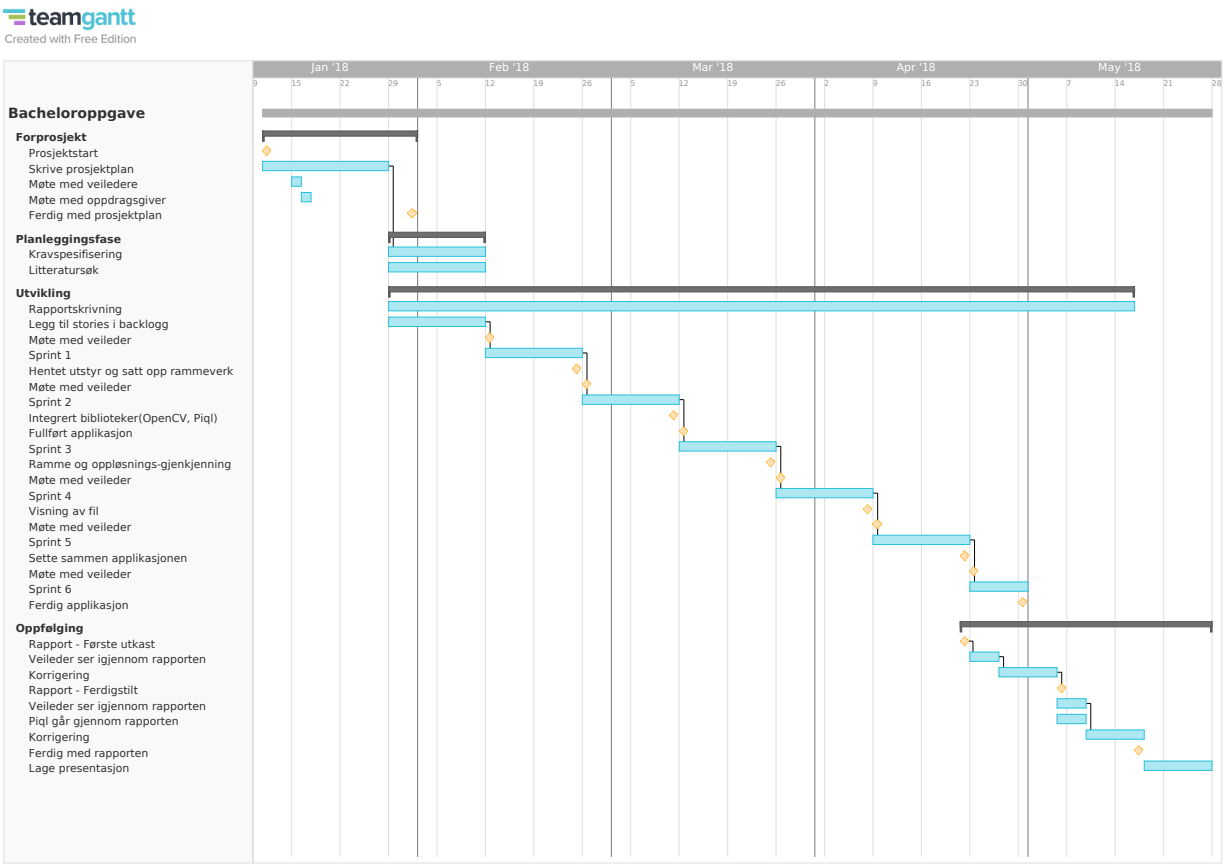
#	Beskrivelse	Tiltak
1	Utstyret kommer ikke	Skrive ut generert kode på papir
2	Gruppen står fast i bildebehandlings delen	Kontakte veiledere
3	Klarer ikke balansere utvikling og rapportskriving	God planlegging
4	Misforståelser rundt produktet	Avklare ønsket med oppdragsgiver
5	Kildekode på avveie	Konfidensialitets- avtale
6	Problemer med å bruke eksisterende kode på Android	Undersøk muligheter til å kjøre på android, evt port kode til android eller skill ut i backend på server.

Tabell 2: Risikotabell

5.3.1 Tiltak for de mest kritiske risikoene

Utstyret kommer ikke
Vi vil få tilgang til å generere den binære dataen som skrives ut på film slik vi kan generere vår egen data. Dermed kan vi skrive det ut på ark og teste med det.

6 Plan for gjennomføring



Figur 1: Gantt-Diagram

Beskrivelse av Gantt skjema
Forprosjektet består av å finne ut hva vi skal drive med. Dette innebærer og få oppgavebeskrivelse ifra oppdragsgiver, bli kjent med veiledere og legge opp en plan for hele prosjektet (prosjektplan). Selv om vi ikke klarer å holde oss til planen gjennom hele perioden, skal den brukes som et utgangspunkt slik at hvis vi sporer av, vet vi når vi er på banen igjen.

I planleggingsfasen vil vi sette opp kravspesifiseringen til applikasjonen. Her skal vi sette opp evt. use-case diagram, objekt orientert design (OOD), systemarkitektur og mer hvis vi ser nødvendighet for dette. Vi vil også gjøre litteratursøk i denne perioden rundt temaet bildebehandling. Vi antar at det finnes en del informasjon her ettersom det er ganske likt en QR-kode applikasjon.

Utviklingsdelen er den mest omfattende delen. Inkludert rapportskrivingen er det denne delen som vi ta mest tid. Rapportskrivingen vil bli jobbet med litt, helt ifra startenm, og vil bli jobbet med mer og mer gjennom hele prosjektet helt til innleveringen 16. mai. Starten vil kun bestå av å legge til "stories" i backloggen for å få en overordnet oversikt over implementasjonen. Deretter begynner sprintene, hver på 2 uker etterfulgt av et møte med veileder. I sprint 5 vil vi begynne å tenkte på å bli ferdig med applikasjonen, her skal vi sette sammen de forskjellige modulene i applikasjonen og passe på at det fungerer. Den siste sprinten (sprint 6) vil bestå av ferdiggjøring, finpussing og bugfixing. Denne sprinten varer kun i en uke ettersom vi er nødt til å få ferdig applikasjonen og fokusere på rapporten.

Oppfølgingsfasen er lagt til for å kvalitetssikre rapporten, her har vi satt oss som krav at rapporten skal være ferdig tidlig slik at vi kan få tilbakemelding på hva som mangler i relativt god tid før innleveringen. Vi har også satt opp mulighet for at Piql skal få se igjennom rapporten før levering ettersom dette var noe de ønsket.

References

[1] "Papirbredden drammen kunnskapspark - piql as." http://www.papirbredden.no/cinevation.aspx. Besøkt 12.01.2018.

[2] I. Sommerville, "Software engineering tenth edition," pp. 43–88, 2016.

F Vitenskapelig analyse

Innlevering 1 - Ingeniørfaglig Systememne

Christian Hådem Håkon Heggholmen
Even Måren Stende

Februar 2018

Foreløpig prosjektmål
Utvikle en applikasjon for lesing av binære data fra film.

Definisjoner
Applikasjon: Program som kjører på Android Smarttelefoner.
Lesing: Ta bilde og prosessere bildet for å hente ut binære data.
Binære data: Data lagret ved bruk av proprietært format utviklet av Piql AS.
Film: Spesiallaget fotosensitiv filmrull for langtidslagring av data.

Objekt
Med denne problemstillingen har vi valg oss *applikasjonen* som objektet.

Teoretiske variable

1. Digitalisering av filmdata.
2. Dekode binær data.

Indikatorer

Teoretisk variabel	Indikator	Validitet
1.	• Barreling. • Pincushioning. • Markør deteksjon. • Homografier. • Oppløsningsverifisering.	Begrenset. • Mange metoder som vi kan nytte oss av. • Vi tar ikke for oss "denoising" , "deblurring", sfærisk aberrasjon, Radiance non-uniformity, kromatisk aberrasjon og mer. • Er det bra nok for å gå videre i applikasjonen. • Kan ikke sørge for perfekt kalibrering.
2.	• Dekode binære data til filer. • Presentere tekst- og bilde filer.	Begrenset • Mangler visning på pdf, og andre filformater.

Konkretiserte indikatorer

Indikator	Konkretisert indikator	Reliabilitet
Barreling og pincushioning	Korrigere barreling og pincushioning ved bruk av kalibrasjonsdata fra camera resectioning i open-cv	Begrenset: Korrigeringen vil føre til tap av data. Mengden data som blir tapt eller feil korrigering vil avhenge av kamera kalibreringen
Markør deteksjon	Bruke eksterne algoritmer for å finne koordinatene til spesialmarkerte hjørner på området som skal dekodes	Begrenset: Eksterne algoritmer. Kan hende den finner feil hjørner.
Homografier	Bruke hjørnekoordinatene og rette opp bildet slik at man ser vinkelrett på det og slik at bildet står vinkelrett	Begrenset: Gjøres av eksterne algoritmer, må ha korrekte hjørnepunkter.
Oppløsningsverifisering	Sjekke kantene på et bilde for å verifisere at ingen data har gått tapt under forvrenging.	Begrenset: Må ha kanter å sjekke. Ikke garantert at man finner feil her.
Dekode binære data til filer	Bruke Piql sitt eksterne bibliotek til å dekode den binære dataen og få ut en "tar-fil".	Begrenset: Vanskelig å modifisere dette biblioteket. Biblioteket er laget til å prosessere bilder fra film scanner med krav til høy presisjon langs en akse i bildet.
Presentere tekst- og bilde filer	"Pakke ut" tar-filen fra piql biblioteket for å vise frem tekstfiler og bilder på formater støttet på Android(PNG, BMP, WEBP, JPEG og GIF). format	Begrenset: Mangler enkelte bildeformater.

Endelig prosjektmål
Utvikle en applikasjon som ved bruk av flere bildebehandlings algoritmer skal digitalisere binære data lagret på film, etterfulgt av å dekode og presentere det på et menneskelig leselig format.

G Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Håkon G. Larsson, pigl

(oppdragsgiver), og

Håkon Heggholmen, Christian Hådem,
Even Måren Stende

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10.01.2018 til 16.05.2018.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Sony George / Marius Pedersen

Oppdragsgivers kontaktperson (navn): Håkon G. Larsson

Student(er) (signatur): Håkon Heggholmen dato 10.01.2018

Christian Halbom dato 10.01.2018

Even Måren Stende dato 10.01.2018

_____ dato _____

Oppdragsgiver (signatur): _____ dato 12.01.2018

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

H Konfidensialitetsavtale

STANDARDMAL

ved avtale om konfidensialitet mellom student og bedrift/ekstern virksomhet i forbindelse med studentens utførelse av oppgave (masteroppgave/prosjektoppgave) i samarbeid med bedrift/ekstern virksomhet, jf. punkt 5 i avtale om oppgave i samarbeid med bedrift/ekstern virksomhet. Malen er fastsatt av Rektor ved NTNU 29.08.2011.

AVTALE
mellom

Student ved NTNU: Even Måren Stende	født: 22.08.1996

Bedrift/ekstern virksomhet: Pigl

om konfidensialitet.

1. Studenten skal utføre oppgave i samarbeid med bedrift/ekstern virksomhet som ledd i sitt studium ved NTNU.

2. Studenten forplikter seg til å bevare taushet om det han/hun får vite om tekniske innretninger og fremgangsmåter samt drifts- og forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde for bedriften/den eksterne virksomheten.
Det er bedriftens ansvar å sørge for å synliggjøre og tydeliggjøre hvilken informasjon dette omfatter.

3. Studenten er forpliktet til å bevare taushet om dette i 5 år regnet fra sluttdato, jf. standardavtale om utføring av oppgave i samarbeid med bedrift/ekstern virksomhet punkt 1.

4. Kravet om konfidensialitet gjelder ikke informasjon som:

- var allment tilgjengelig da den ble mottatt
- ble mottatt lovlig fra tredjeperson uten avtale om taushetsplikt
- ble utviklet av studenten uavhengig av mottatt informasjon
- partene er forpliktet til å gi opplysninger om i samsvar med lov eller forskrift eller etter pålegg fra offentlig myndighet

Even Måren Stende, NTNU Gjøvik, 15.01.2018
.....
sted, dato student

Piql AS
Grønland 56
N-3045 Drammen, NORWAY
Org nr: NO 984359403

Dram 27/4

.....

sted, dato

for bedrift/ekstern virksomhet
stempel og signatur

STANDARDMAL

ved avtale om konfidensialitet mellom student og bedrift/ekstern virksomhet i forbindelse med studentens utførelse av oppgave (masteroppgave/prosjektoppgave) i samarbeid med bedrift/ekstern virksomhet, jf. punkt 5 i avtale om oppgave i samarbeid med bedrift/ekstern virksomhet. Malen er fastsatt av Rektor ved NTNU 29.08.2011.

AVTALE
mellom

Student ved NTNU: Håkon Heggholmen	født: 13.02.94

Bedrift/ekstern virksomhet: Pigl

om konfidensialitet.

1. Studenten skal utføre oppgave i samarbeid med bedrift/ekstern virksomhet som ledd i sitt studium ved NTNU.

2. Studenten forplikter seg til å bevare taushet om det han/hun får vite om tekniske innretninger og fremgangsmåter samt drifts- og forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde for bedriften/den eksterne virksomheten.
Det er bedriftens ansvar å sørge for å synliggjøre og tydeliggjøre hvilken informasjon dette omfatter.

3. Studenten er forpliktet til å bevare taushet om dette i 5 år regnet fra sluttdato, jf. standardavtale om utføring av oppgave i samarbeid med bedrift/ekstern virksomhet punkt 1.

4. Kravet om konfidensialitet gjelder ikke informasjon som:

- var allment tilgjengelig da den ble mottatt
- ble mottatt lovlig fra tredjeperson uten avtale om taushetsplikt
- ble utviklet av studenten uavhengig av mottatt informasjon
- partene er forpliktet til å gi opplysninger om i samsvar med lov eller forskrift eller etter pålegg fra offentlig myndighet

Håkon Heggholmen, NTNU Gjøvik, 15.01.2018

sted, dato student

Drammen 27/4

Piql AS
Grønland 56
N-3045 Drammen, NORWAY
Org.no: NO 984359403

sted, dato

for bedrift/ekstern virksomhet
stempel og signatur

STANDARDMAL

ved avtale om konfidensialitet mellom student og bedrift/ekstern virksomhet i forbindelse med studentens utførelse av oppgave (masteroppgave/prosjektoppgave) i samarbeid med bedrift/ekstern virksomhet, jf. punkt 5 i avtale om oppgave i samarbeid med bedrift/ekstern virksomhet. Malen er fastsatt av Rektor ved NTNU 29.08.2011.

AVTALE
mellom

Student ved NTNU: Christian Seeberg Hådem	født: 26. sep. 1996

Bedrift/ekstern virksomhet: Pigl

om konfidensialitet.

1. Studenten skal utføre oppgave i samarbeid med bedrift/ekstern virksomhet som ledd i sitt studium ved NTNU.

2. Studenten forplikter seg til å bevare taushet om det han/hun får vite om tekniske innretninger og fremgangsmåter samt drifts- og forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde for bedriften/den eksterne virksomheten.
Det er bedriftens ansvar å sørge for å synliggjøre og tydeliggjøre hvilken informasjon dette omfatter.

3. Studenten er forpliktet til å bevare taushet om dette i 5 år regnet fra sluttdato, jf. standardavtale om utføring av oppgave i samarbeid med bedrift/ekstern virksomhet punkt 1.

4. Kravet om konfidensialitet gjelder ikke informasjon som:

- var allment tilgjengelig da den ble mottatt
- ble mottatt lovlig fra tredjeperson uten avtale om taushetsplikt
- ble utviklet av studenten uavhengig av mottatt informasjon
- partene er forpliktet til å gi opplysninger om i samsvar med lov eller forskrift eller etter pålegg fra offentlig myndighet

Christian Hådem, NTNU Gjøvik, 15.01.2015

sted, dato student

Drammen 27/4

Pigl AS
Grønland 56
N-3045 Drammen, NORWAY
Org.no: NO 984359403

sted, dato

for bedrift/ekstern virksomhet
stempel og signatur

I Brukertest epost

SV: Håkon Heggholmen wants to share the file app-debug.apk with you

Håkon Larsson <hakon.larsson@piql.com>

ti 03.04.2018 13:54

Innboks

Til:Håkon Heggholmen <haakheg@stud.ntnu.no>;

Supert. Stå på.

\H

Fra: Håkon Heggholmen <haakheg@stud.ntnu.no>
Sendt: tirsdag 3. april 2018 13:41
Til: Håkon Larsson <hakon.larsson@piql.com>
Emne: SV: Håkon Heggholmen wants to share the file app-debug.apk with you

Kræsj under kalibrering kan skyldes at bildet inneholder mye svart og tlf henger seg opp ettersom den finner mønsteret over alt. Kan se på termcriteria for å få den til å "gi opp" søket fortere.
0.3 FPS skyldes undistortion og ramme-søk, noe som rett og slett er krevende operasjoner. Skal også se om det er noen måte å få mobilen til å ikke gå i dvale, men om det ikke finnes kan det endres i innstillingene på tlf. På mobilen vi bruker for utvikling har vi ca 0.5 FPS på høyeste oppløsning. Vi planlegger også å legge inn en "Del" knapp, slik at ikke støttede filer som pdf osv kan åpnes i en annen applikasjon.

Fra: Håkon Larsson <hakon.larsson@piql.com>
Sendt: 3. april 2018 13:04:28
Til: Håkon Heggholmen
Emne: SV: Håkon Heggholmen wants to share the file app-debug.apk with you

Hei da har jeg testet app'n. Etter en del om og men klarte jeg å dekode en ramme. Det var gøy.

Men bruker opplevelsen av app'n er at den er veldig treg. Om det skyldes SW, HW eller boxing lib vet jeg ikke.

Her er hva jeg opplevde:

1. Den krasjet et par ganger under kalibrering
2. Det tok så lang tid med kalibrering at app'n gikk i dvale (svart skjerm) og etter at jeg vekket den opp startet kalibrering på nytt.
3. Det er mulig at min telefon er litt treg for bilde preview hakket veldig (0.3 FPS)

\Håkon