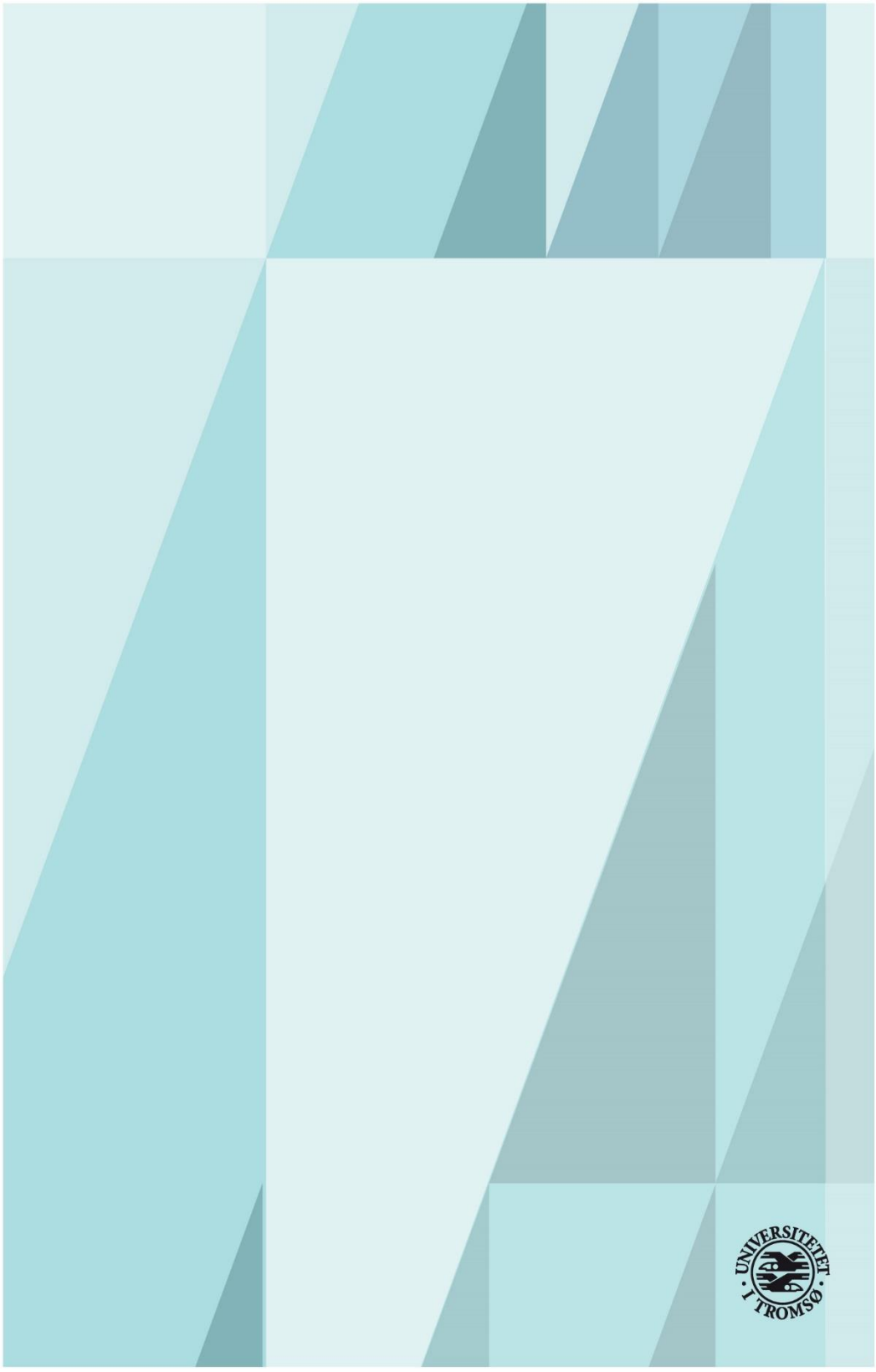
UiT

NORGES ARKTISKE UNIVERSITET

PROSJEKTRAPPORT BILDEBEHANDLING

Fakultet for ingeniørvitenskap og teknologi
ITE1914, 2019H
Kristoffer Johan Garmann



1 INNLEDNING

1.1 PROBLEMSTILLING

Denne rapporten beskriver min tilnærming til å løse oppgave 9 i ITE1914 Signal- og Bildebehandling:

Lag et system som kan lese av en analog klokke med et kamera og skrive ut tiden "digitalt". Nøyaktighet = 1 minutt.

Eneste krav til løsningen er at bildet som skal behandles blir tatt av et digitalkamera.

1.2 PRESENTASJON

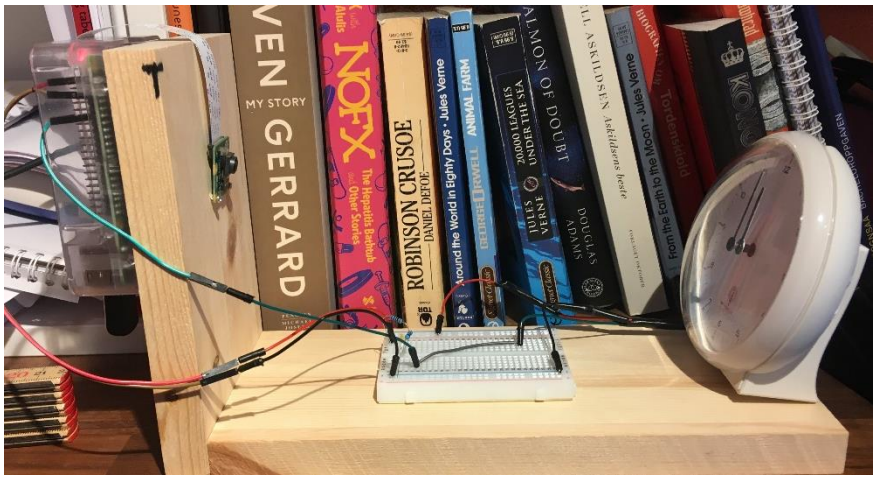
I tillegg til å gjennomføre prosjektet skal det også presenteres. Det oppmuntres til en sanntidsdemo med forklaring av hvert steg i bildebehandlingsprosessen, og dette har jeg lagt ekstra vekt på i løpet av prosjektet.

2 METODE

Min egen målsetning for oppgaven er å lage en fast måleoppstilling som kan automatiseres og fjernstyres. I store deler av prosjektperioden er jeg ikke hjemme på mitt eget kontor, og vil ha behov for fjerntilgang for å få jobben gjort. Jeg er også avhengig av å skrive en robust kode som kan takle forskjellige lysnivåer.

2.1 UTSTYR

Jeg har brukt en Raspberry Pi 2 B+ ettkorts datamaskin som jeg har liggende som hjernen i operasjonen. Denne kan man koble til et rimelig enkelt kamera som kan styres direkte og den har en rekke GPIO-pinner jeg kan styre som jeg vil. I tillegg har jeg gått til innkjøp av ei enkel klokke. Dette har jeg snekret sammen i en måleoppstilling som peker kameraet direkte på klokka som vist i Figur 1. Klokka er ikke helt i fokus, men det er nok ikke viktig for å løse oppgaven. Ledningene mellom datamaskinen og klokka er for å både gi strøm til klokka og for å utløse en reset-funksjon på klokka som får viserne til å snurre rundt klokka en gang slik at jeg kan teste koden min på et tilfeldig «tidspunkt».



Figur 1 Måleoppstilling

Klokka er en billig variant, men den stilles automatisk etter DCF77-signalet. DCF77 er et langbølge radiosignal som sendes ut fra Tyskland og kan brukes til å stille klokker automatisk. Jeg skal ikke gå så mye inn på akkurat dette, men den opprinnelige tanken med å koble sammen klokka og datamaskinen var å hente ut dette signalet i den hensikt å sammenligne resultatet fra min kode med hva klokka selv «trodde» den var. Jeg fant signalet på kretskortet til klokka men dekningen var for dårlig til å lese dette av pålitelig. Klokka stilte seg selv flere ganger, men ofte bare på nattestid når jeg ikke så på. Det var ikke brukbart til kontinuerlig avlesning.

2.2 PROGRAMVARE

For å løse oppgaven har jeg i hovedsak programmert i Python. Python er ikke det jeg har mest erfaring med, men virker veldig lett å komme i gang med dette på en Raspberry Pi så jeg har tatt den utfordringen. Noen utfordringer i prosjektet blir for tunge for Python og da har jeg gått over til C++ som ligger på et mye lavere nivå i maskinen og spesielt minneoperasjoner går derfor mye fortere. Kompleksiteten øker, men jeg har til gjengjeld mer erfaring med C/C++ så programmeringen flyter lettere.

Når jeg har fått måleoppstillingen i gang så langt at jeg kan ta et bilde fra kommandolinjen har jeg brukt Adobe Photoshop til å teste meg manuelt frem til hvilken metode jeg vil bruke for å løse oppgaven. Operasjonene jeg gjør manuelt i Photoshop finner jeg igjen i et åpent bibliotek for bildebehandling som heter OpenCV, som er tilgjengelig for både Python og C++.

Til slutt har jeg lagd et enkelt web-grensesnitt for den endelige programvaren ved hjelp av Flask-rammeverket som er tilgjengelig i Python. Dette er ikke en del av oppgaven men jeg har tatt det med da det er en viktig del av presentasjonen.

2.3 FREMGANGSMÅTE

For å finne ut hvor mye klokka er ut i fra et bilde har jeg delt prosessen inn i to hoveddeler. Først må jeg finne ut hvor klokka er i bildet, deretter må jeg tolke klokka på et vis. Det finnes flere løsninger og jeg har utforsket noen av disse.

2.3.1 Finne klokka

For å finne klokka i bildet har jeg brukt to forskjellige metoder der en er veldig enkel og den andre er mer komplisert. Den enkle metoden er å forhåndsdefinere hvor klokka befinner seg. Klokka står på samme plass i hvert eneste bilde fordi både kamera og klokke er fastmontert. Den kompliserte metoden er å bruke en metode som kalles Hough Circle Transform, som vil returnere de delene av bildet som ligner på sirkler. Klokka er stort sett den tydeligste sirkelen i bildet. Jeg har valgt å bruke Hough Circle Transform for å klippe ut kun klokkeskiva, men dersom metoden feiler vil programmet falle tilbake på det forhåndsdefinerte klokkeområdet i bildet.

Hough Circle Transform gå i korte trekk ut på å finne konturene i et bilde ved hjelp av Canny Edge Detection, og deretter teste sirkler innenfor gitte parametere på disse punktene. De punktene der sirklene sammenfaller peker på origo til en mulig sirkel i bildet. Som eksempel kan vi tenke oss en sirkel med radius R. Dersom man går gjennom hvert enkelt punkt på denne sirkelen og tegner en ny sirkel med radius R vil alle disse sirklene overlappe i den opprinnelige sirkelens origo. Dette er implementert som funksjon i OpenCV-biblioteket så jeg har heldigvis sluppet å programmere det.

Canny Edge Detection er også en funksjon som er implementert i OpenCV som er en del av Hough Circle Transform og Hough Line Transform som jeg kommer tilbake til. Jeg kun brukt denne funksjonen for å vise dette mellomsteget i Hough-transformene.

Jeg bruker altså kun Hough Circle transform for å finne den *mest utpregede sirkelen som har radius over en fjerdedel av bildets lengde.*

I Figur 2 under viser jeg delprosessene fra jeg tar bilde til jeg har funnet denne sirkelen jeg er ute etter.

Figur 2 Prosessen for å finne klokka fra bildetaking til Hough Circle Transform

Når klokka er funnet klipper jeg ut firkanten rundt sirkelen samtidig som jeg maskerer ut sirkelen slik at bakgrunnen blir helt hvit. Da står vi igjen med et bilde som ligner på Figur 3 under. Nå er hele bakgrunnen klippet bort og vi står igjen med den interessante delen av bilde. Uansett hvilken metode som bruke er det er bilde som dette som er utgangspunktet for neste del av prosessen.

Figur 3 Typisk resultat av første steg

2.3.2 Finne ut hvor mye klokka er

Uansett hvilken metode man bruker for å tolk klokkeslettet ut i fra viserne på klokka er det nødvendig å isolere viserne fra resten av bildet. Med svarte visere på hvit bakgrunn er det relativt enkelt å gjøre en terskling for å komme frem til et binært bilde der viserne vil være den største formen i bildet. Før tersklingen er det likevel lurt å fjerne støy med et gaussisk filter. Dette gjør jeg med en 5x5-maske. Hensikten med filteret er at bildet skal få jevnere overganger og dermed klarere bilde og mindre flekker etter terskling.

2.3.2.1 Adaptiv terskling

Når jeg skal bruke terskling på bildet er det vanskelig å sette en fast terskelverdi som vil fungere under forskjellige lysforhold. For å løse dette har jeg laget et lite sett med regler som setter terskelverdien. Ved å analysere histogrammene til bilder tatt ved forskjellige lysstyrker ser jeg at det finnes veldig mye hvitt i bildet. Dette er fordi hjørnene også er en del av bildet. Dette er ikke interessant, så jeg kutter ut de ti høyeste verdiene(intensitet>245) fra histogrammet. Deretter legger jeg merke til at de høyeste verdiene samler seg rundt intensiteten til viserne som vist i Figur 4. Ut i fra dette har jeg lagd en funksjon som tar den intensiteten med høyest verdi i histogrammet og setter terskelen til 20 verdier under denne. Dette fungerer veldig bra for alle bilder som ikke er veldig lyse eller veldig mørke. Jeg har vurderte å gjøre denne offseten variabel etter lysforhold også, men da den faste verdien fungerte i alle de normale testforholdene mine bestemte jeg at det var nok av det gode. Uansett så ligger terskelen alltid litt under der det er forventet at viserne befinner seg så de kommer alltid med.

Figur 4 Histogrammet til Figur 3

Når tersklingen er ferdig inverteres bildet av praktiske årsaker. Den delen av bildet jeg er interessert i blir hvitt(1) og alt som er uinteressant blir svart(0). Dette er illustrert i Figur 5 under.

Figur 5 Eksempel på bilde etter terskling og invertering

Dersom det fremdeles finnes små flekker igjen på dette bildet blir de fjernet gjennom åpning. En åpning er gjentatt erosjon/dilasjon av bildet. Dette gjøres fordi det vil gjøre neste steg betraktelig enklere for datamaskinen. Det skjer også i noen tilfeller at minuttviseren overlapper et tall på klokka, og disse blir som regel adskilt ved åpning.

2.3.2.2 Connected component labeling

Når man ser på bildet i Figur 5 vil jeg si det er rimelig å anta at den største figuren i bildet er viserne. For å trekke ut viserne fra bildet vil jeg bruke **connected component labeling(CCL)** gjennom two-pass metoden som er først beskrevet i 1976 (Hoshen & Kopelman, 1976).

Når man skal se på hver enkelt piksel i hele bildet og deres naboer på en Raspberry Pi 2 blir det nødvendig å bytte ut Python med C++ for å ikke bruke for lang tid.

Målet er å gi hver enkelt «figur» i bildet en egen id som alle pikslene i figuren deler. Deretter returnerer man kun den figuren med flest piksler. Dette gjøres ved å først gå gjennom alle piksler fra høyre mot venstre og fra topp til bunn. De markeres med en verdi basert på nabopikslene over og til venstre for den aktuelle pikslen. Dette gjøres etter følgende regler:

1. Dersom kun en av naboene har en verdi, eller naboene har samme verdi, vil pikselen få denne verdien.
2. Dersom begge naboene har en verdi vil pikselen få den laveste verdien og naboskapet lagres i en tabell.
3. Dersom ingen av naboene har en verdi vil denne pikselen få en ny verdi.

Deretter sorteres naboskapene slik at hver eneste verdi peker på laveste verdi den er i kontakt med.

For å illustrere bruker jeg en matrise for å representere et lite binært bilde i Figur 6 under.

0	0	0	0	0	1	0	0
0	1	0	0	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	0
0	1	1	0	0	0	1	0
0	1	0	0	0	1	1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Figur 6 Binær representasjon av et bilde

Etter CCL vil matrisen se ut som i Figur 7:

0	0	0	0	0	1	0	0
0	2	0	0	3	1	0	0
0	0	0	4	3	0	0	0
0	0	0	4	0	0	0	0
0	5	5	0	0	0	6	0
0	5	0	0	0	7	6	0
0	0	8	0	0	0	0	0
0	0	0	0	0	0	0	0

Figur 7 Første gjennomgang av matrisen, naboskapene er markert

Ut i fra første gjennomgang vil man stå igjen med følgende naboskap og reduseringer:

3→1	
4→3	4→1
7→6	

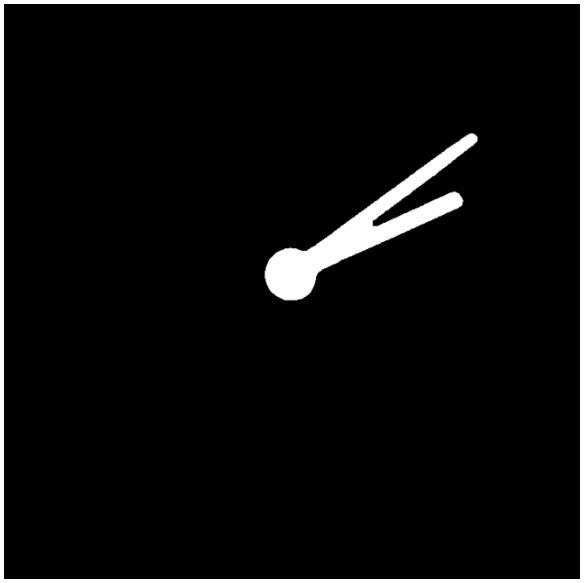
Det endelige bildet vil se ut som i Figur 8.

0	0	0	0	0	1	0	0
0	2	0	0	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	0
0	5	5	0	0	0	6	0
0	5	0	0	0	6	6	0
0	0	8	0	0	0	0	0
0	0	0	0	0	0	0	0

Figur 8 Markerte figurer

Dette er kun en enkel forklaring av prinsippet bak figurinndelingen, men det gjelder også i de større bildene brukt i dette prosjektet selv om tabellen med naboskap blir betraktelig større med flere nivåer enn i akkurat dette eksempelet.

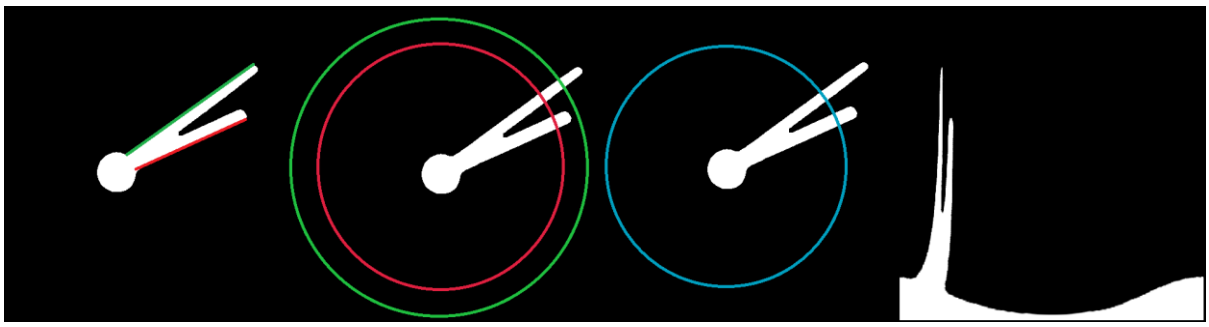
Når vi har kommet så langt i prosessen vil bilde vi jobber med se ut som i Figur 9 under.



Figur 9 Bilde som skal inkludere kun visere

Nå jobber vi med et bilde som kun inneholder nyttig informasjon. Alle tallene og støy er borte, det er kun viserne som gjenstår. Vi vet at klokka er 12 på toppen av bildet og 6 på bunnen. Hvordan finner man ut hvilken vei viserne peker?

For å komme videre har jeg vurdert flere metoder. Man kan lage en ring for hver viser og se hvor på ringen viseren krysser. Man kan lage en ring for begge viserne og skille de fra hverandre på visernes tykkelse. Man kan prøve å finne vinkelen på viserne og skille disse på vinkel og lengde. Og så kan man også gjøre det som jeg har kalt for depolarisering. Det vil si å transformere bildematrisen fra polare til rektangulære koordinater. Konseptene er illustrert i Figur 10 under.



Figur 10 Linjedeteksjon, dobbelsirkel, enkelsirkel, depolarisering

Under hele prosjektet har jeg tenkt at løsningen bør være så generisk som mulig, slik at man for eksempel kan bruke en annen klokke, eller flytte klokken i bildet. Jeg har derfor forkastet metodene med visersirkler da disse er for avhengige av klokka.

I utgangspunktet bestemte jeg meg for å depolarisere klokka for å lese av resultatet som et signal med en eller to topper, men med litt ekstra tid tilgjengelig gjennomførte jeg begge metodene.

2.3.2.3 Depolarisering

Hovedmetoden min går ut på å brette ut klokka og gjøre bildet om til et signal som kan tolkes. Første steg fra bildet til høyre i Figur 10 er å fylle ut de konkave områdene under viserne slike at jeg står igjen med et kontinuerlig signal fra kl 12 til kl 12. Deretter finner jeg maksverdien i dette

signalet og kutter alt som er under en halvparten av dette. Da vet jeg at jeg kun står igjen med viserinformasjon som i Figur 11.

Figur 11 Viserinformasjon

Ut i fra dette kan man bestemme hvor mye klokka er hvis man vet hvilken posisjon i signalet som tilsvarer hvilken vinkel på urskiva. I denne metoden finner jeg signaltopper og deres høyder. De må være tydelig separate for at de skal telles som to forskjellige topper. Timeviseren rundes ned til nærmeste time og minuttviseren rundes ned til nærmeste minutt. Det oppstå noen unntak som enten kan være reelle eller feilsituasjoner:

- Dersom man finner flere visere har man en feilsituasjon og bildet er mest sannsynlig uleselig etter tersklingen.
- Dersom man kun finner én viser kan det hende at viserne peker i samme retning. Da må programmet utføre en sjekk på om dette er reelt eller ikke. Denne ene viseren må peke i en reell retning for at resultatet skal aksepteres. Dersom klokka er 12 peker begge viserne oppover, men de neste gangene viserne peker i samme retning vil ikke klokka peke direkte på et tall. Jeg har kommet frem til at det går 12/11 timer mellom hver gang. Derfor bør den ene viseren peke på n/11 av en time rundt klokka dersom resultatet skal være reelt. Dette er tatt hensyn til i koden. Dersom dette ikke er tilfelle er det sannsynlig at tersklingen eller åpningen har kuttet av en viser. **Dette er avhengig av om timeviseren beveger seg kontinuerlig eller hopper en time av gangen!**
- Dersom man ikke finner noen visere har man også en feilsituasjon.

Dette er hovedløsningen i prosjektet.

2.3.2.4 Hough Line Transform

Man kan også finne klokkeslett ved å analysere linjer i bildet. Jeg har implementert denne metoden gjennom Hough Line Transform-funksjonen i OpenCV. Jeg går ikke noe nærmere inn på nøyaktig hvordan Hough Line Transform fungerer (det er en helt egen oppgave spør du meg).

Men det jeg bruker den til er å finne de lengste linjene i bildet som er over en viss lengde. Deretter ser jeg etter de to lengste linjene som ikke har samme vinkel. Vinkelen jeg regner ut tar hensyn til retning så selv om viserne peker i motsatt retning vil vinkelen være 180 grader fra hverandre. Her gjelder samme regler som ved depolarisering, dersom man kun finner en viser må retningen på denne peke sånn ca i retning av n/11 av en time.

3 BEARBEIDING

Jeg har kun testet programmet mitt «manuelt» og det fungerer i alle tilfeller der belysningen er jevn og tilstrekkelig. Det fungerer også godt i dårlig belysning og sterk belysning så lenge denne er jevn og ikke produserer gjenskinn som overlapper viserne.

Jeg hadde håpet å koble DCF77-signalet fra klokka inn i programmet slik at jeg kunne gjort en automatisert test over tid slik at jeg hadde et stort utvalg tester å lene meg på. Denne løsningen brukte jeg en del tid på å finne ut at ikke fungerte så jeg gav opp automatiserte tester. Dette kunne jeg løst på en annen måte ved å synkronisere klokken med maskinens klokka dersom jeg hadde hatt bedre tid.

I endelig versjon har jeg vel gjennomført rundt 100 tester manuelt under normale forhold som alle er innenfor kravet på 1 minutt.

Noen utfordringer oppstår fordi klokka mi har en minuttviser som overlapper tall og noen symboler på urskiva. Disse unøyaktighetene ser jeg i resultatet «under panseret», men metoden min sørger for at disse er innenfor kravet på 1 minutt. Med en «ren» klokke ville resultatet vært mer nøyaktig.

Ved å holde fokus på en robust løsning har jeg lært mye mer enn hvis jeg hadde valgt en enkel løsning. Jeg kunne klippet ut klokka av bildet manuelt og belyst den slik at den alltid ser lik ut og fått en nøyaktig resultat fra dette hver gang. Da hadde jeg aldri kommet inn på Hough-transformene eller stilt verdier relative til hverandre og relative til bildet. Det blir mer arbeid, men så lenge det er lærerikt og ikke minst underholdene er ikke det et problem.

Dersom jeg skulle jobbet videre med akkurat dette prosjektet ville jeg undersøkt muligheten for å kunne flytte rundt på klokka i bildet og fremdeles få samme resultat. Jeg kunne også tenkt meg å se om jeg kunne funnet klokka og fulgt viserne direkte på videostreamen med Hough Line Transform i realtime. Jeg tror det er mulig, men da må jeg kanskje koble meg på noe mer kraftig enn en Raspberry Pi 2.

- Vedlagt er to gjennomførte tester av de forskjellige metodene(Vedlegg 1 & 2)
- All kode ligger på https://github.com/onklgarmann/clockwatch

4 OPPSUMMERING

I denne rapporten har jeg dokumentert arbeidet jeg har gjort for å kunne lese av en analog klokke ved hjelp av et digitalkamera og litt programmering. Programmet fungerer svært godt innenfor rimelighetens grenser. Jeg har lagt vekt på robusthet og presentasjon av prosjektet.

Resultatet ble svært bra til tross for at jeg ikke nådde alle mine egne målsetninger. Skulle gjerne brukt mer tid på prosjektet men tiden og klokka har sine begrensninger.

5 VEDLEGG

1. Clockwatch.pdf
2. Clockwatch_hough.pdf

6 REFERANSER

Hoshen, J., & Kopelman, R. (1976). Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. *Physical Review*.