NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

TTT4280 – Labrapport 2

Tittel: Optikk Lab

Skrevet av: Freider Engstrøm Fløan og Theodor Lembrecht Qvist

Gruppe: 44

Dato: 25. april 2024

Figur 1: Dalle-3.

Sammendrag

Innhold

1 Innledning **1**

2 Teori **2**
2.1 Biooptikk . . . 2
2.1.1 Lyspropagering i Vev . . . 2
2.1.2 Absorpsjon og Spredning . . . 2
2.2 Fotontransport . . . 3
2.2.1 Matematisk Modellering av Fotontransport . . . 3
2.3 Reflektans og Transmittans . . . 5
2.3.1 Reflektans . . . 5
2.3.2 Transmittans . . . 5
2.4 Regning på Pulsutslag . . . 6
2.4.1 Matematisk Modellering av Pulsutslag . . . 6
2.4.2 Analyse av Pulsamplitude . . . 6
2.5 Signalbehandling - FFT . . . 6
2.6 Signal-to-Noise Ratio i Pulssammenheng . . . 7
2.6.1 Definisjon av SNR . . . 7
2.6.2 Betydning av Høy SNR . . . 7
2.6.3 FFT-basert Tilnærming til SNR . . . 7
2.6.4 Vindusfunksjoner . . . 8
2.6.5 Zero-padding i Spektralanalyse . . . 8
2.7 Relevante labforberedelser . . . 9
2.7.1 Penetrasjonsdybde i menneskevev . . . 9
2.7.2 Prosentandel av lys gjennom finger . . . 9
2.7.3 Reflektans i menneskevev . . . 9
2.7.4 Kontrast . . . 10
2.7.5 Egnethet til pulsmåling . . . 11

3 Systemoppsett **12**
3.1 Utstyrsliste . . . 12
3.2 Fysisk oppsett . . . 12
3.3 Signalbehandling . . . 13
3.3.1 FFT . . . 13
3.3.2 SNR . . . 13

4 Resultater og diskusjon **14**

5 Konklusjon **17**

Referanser **18**

A Vedlegg A **19**

B Vedlegg B **21**

C Vedlegg C **22**

D Vedlegg D **23**

1 Innledning

I denne rapporten presenterer vi resultatene fra laboratorieøvelsen i TTT4280 - Sensorer og Instrumentering vedrørende bruk av optiske metoder for å studere biologiske prosesser. Målet med laboppgavene er å estimere pulsen til en person med hjelp av optiske målinger.

Optiske teknikker er viktig i moderne biomedisinske anvendelser, fra diagnostikk til behandling. I denne øvelsen har vi fokusert på hvordan lys interagerer med menneskelig vev, og hvordan disse interaksjonene kan utnyttes for å utlede viktig biologisk informasjon som blodoksidasjonsnivåer og puls.

Denne labøvelsen gir innsikt i kompleksiteten og relevansen av biooptikk i et realistisk anvendelsesområde. Resultatene fra våre eksperimenter understreker betydningen av presisjon i både oppsett og analyse for å oppnå pålitelige målinger, og diskuteres detaljert gjennom de etterfølgende seksjonene av rapporten.

2 Teori

2.1 Biooptikk

Biooptikk omhandler studiet av biologiske materialer ved bruk av optiske metoder. Dette feltet utnytter de unike egenskapene til lysinteraksjoner med biologiske vev for å måle og analysere biologiske prosesser på en ikke-invasiv måte.[1]

2.1.1 Lyspropagering i Vev

Når lys samhandler med vev, kan det absorberes, spres, transmitteres eller reflekteres. Lysets oppførsel i vev avhenger hovedsakelig av vevets optiske egenskaper, som inkluderer absorpsjons- og spredningskoeffisienter. Beer-Lamberts lov, som tradisjonelt brukes til å beskrive lysabsorpsjon i klare medier, er ofte utilstrekkelig for komplekse biologiske vev på grunn av den høye spredningen i disse materialene. I stedet brukes modifiserte modeller som diffusjonsapproximasjonen for bedre å ta hensyn til spredning og absorpsjon samtidig.[2]

$$I(z) = I_0 e^{-\mu_a z} \tag{1}$$

Der I_0 er den innkommende lysintensiteten, z er dybden lys har bevegtet seg, og μ_a er absorpsjonkoeffisienten.

2.1.2 Absorpsjon og Spredning

Absorpsjon i vev skyldes primært kromoforer som hemoglobin i blodet, som har distinkte absorpsjonsspektra avhengig av dets oksygeneringstilstand. Forskjellen i absorpsjon av oksyhemoglobin og deoksyhemoglobin muliggjør måling av blodoxygenasjonsnivåer og pulsrate gjennom teknikker som pulsoximetri.[2]

Absorpsjonskoeffisienten μ_a er gitt ved,

$$\mu_a = C\epsilon \tag{2}$$

der C er molarkonsentrasjonen og ϵ er den molare ekstinksjonskoeffisienten. Molarkonsentrasjonen har benevningen [M], og den molare ekstinksjonskoeffisienten har benevningen $[(cm^{-1})/M]$. Absorpsjonskoeffisienten beskriver sannsynligheten for absorpsjon per lengdeenhet $[cm^{-1}]$.

Spredning i vev forårsakes av variasjoner i brytningsindeksen på mikroskopisk skala, slik som cellebegrensninger og intracellulære strukturer. [3]

Absorpsjonskoeffisienten til blod kan variere basert på blodets oksygeninnhold. Inne i blodet finner vi hemoglobin, et molekyl som kan binde oksygen. Hemoglobin består av fire heme-

grupper, og hver av disse gruppene kan binde seg til ett oksygenmolekyl. Når oksygen bindes til heme-gruppen, endres molekylets struktur, inkludert avstanden mellom bindingene. Denne strukturendringen påvirker blodets absorpsjonsegenskaper og farge, noe som reflekteres i dets absorpsjonsspekter. [3] Dette leder oss til følgende karakteristikk av blodets absorpsjonskoeffisient:

$$\mu_{a_{blod}} = s \cdot \mu_{a_{oks}} + \mu_{a_{deoks}} \cdot (1 - s) \tag{3}$$

der $\mu_{a_{oks}}$ og $\mu_{a_{deoks}}$er absorpsjonskoeffisientene for henholdsvis oksygenrikt og oksygenfattig blod. Disse koeffisientene varierer med lysbølgelengden og kan oppslås i tabeller. Oksidasjonsforholdet s beskriver hvor stor andel av blodet som er oksygenrikt. Gitt at absorpsjonskoeffisientene varierer med oksidasjonsnivået, vil lys med samme bølgelengde absorbere forskjellig avhengig av oksygeneringsgraden. Det finnes også bølgelengder hvor oksidasjonsgraden ikke påvirker absorpsjonen, kjent som isosbestiske punkter. [2]

Spredningskoeffisienten μ_s er gitt ved,

$$\mu_s = \sigma_s N \tag{4}$$

der σ_s er sprednings-tverrsnittet til en individuell partikkel, med enheten $[cm^2]$, og N er N er konsentrasjonen av partikler, med enheten $[cm^{-3}]$. Spredningskoeffisienten beskriver sannsynligheten for spredning per lengdeenhet$[cm^-1]$.

2.2 Fotontransport

Fotontransport beskriver overføringen av lysenergi gjennom et medium, som for eksempel biologiske vev. Dette konseptet er viktig for å forstå hvordan lys interagerer med biologisk materiale i diverse biooptiske teknikker. Når lys inntreffer biologisk vev, vil det bli påvirket av mediets optiske egenskaper gjennom absorpsjon og spredning.

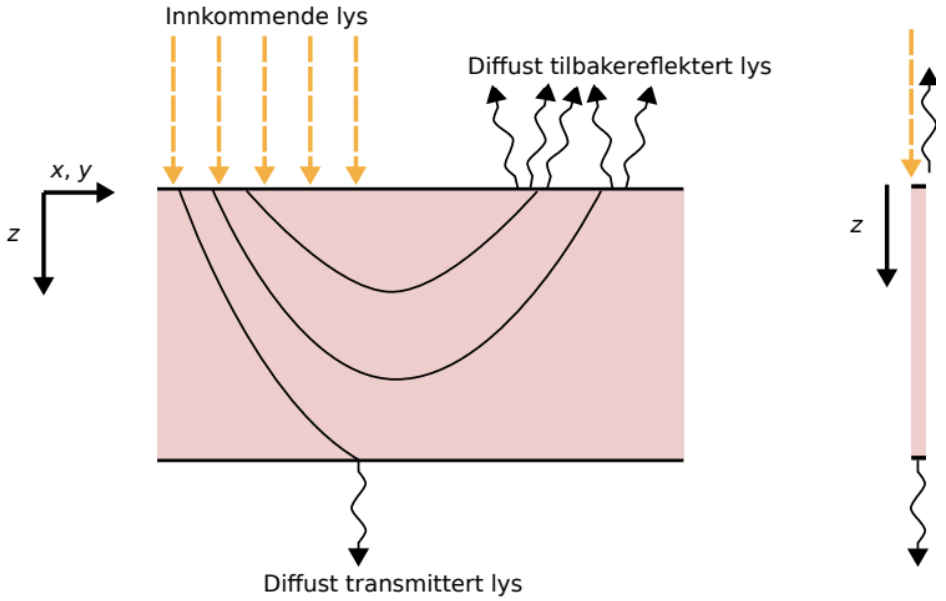
2.2.1 Matematisk Modellering av Fotontransport

For å modellere fotontransport i vev anvendes diffusjonslikningen, som er en fundamental ligning i transportteorien for fotoner. Denne andre ordens partielle differensiallikningen beskriver hvordan fluensen, ϕ, som representerer fotonintensiteten, distribueres i vevet:

$$\frac{\partial \phi(\vec{r}, t)}{c\partial t} + \mu_a \phi(\vec{r}, t) - D\nabla^2 \phi(\vec{r}, t) = S(\vec{r}, t) \tag{5}$$

Her er ϕ fluensen, μ_a er absorpsjonskoeffisienten, D er diffusjonskoeffisienten som avhenger av både sprednings- og absorpsjonskoeffisientene, S representerer lyskilden, og $\vec{r}$ og t angir posisjon og tid.

Ligningen gir innsikt i hvordan fluensraten endres over tid på en spesifikk posisjon $\vec{r}$ i vevet ved et gitt tidspunkt t. Denne endringen avhenger av hvor mye fotonene absorberes (μ_a), der høyere absorpsjonsnivå resulterer i færre fotoner tilgjengelig for diffusjon og dermed en mindre endring i fluensraten. Videre påvirkes endringen av diffusjonskonstanten D, definert som $D = \frac{1}{3}(\mu_s' + \mu_a)$, der μ_s' representerer den reduserte spredningskoeffisienten. Denne reduserte spredningskoeffisienten antas å være isotropisk spredt, og er mer inngående beskrevet i referanse [4].



Figur 2: Fotontransportmodellen. Hentet fra [2].

Med noen forenklinger, som å anta at fluensraten er tiduavhengig, at vevet er homogent, og at lyset som treffer huden er en planbølge, blir fluensraten kun avhengig av z-komponenten. Dette representerer dybden i vevet, som illustrert i Figur 2. Ytterligere, ved å anta at lyskilden $S(z)$ er en Dirac-delta-funksjon, $S(z) = \delta(z)$, kan diffusjonslikningen forenkles betydelig. Selv om dette reduserer nøyaktigheten og presisjonen av beregningene, gir det en tilstrekkelig tilnærming for dette prosjektet for å estimere pulsen. Den forenklede diffusjonslikningen som da beskriver lystransporten blir:

$$\mu_a \phi(z) - D\frac{d^2\phi(z)}{dz^2} = 0 \tag{6}$$

For å løse denne likningen har vi valgt å ignorere grensebetingelser mellom hud og luft og andre faktorer som lagdeling av huden og Snells lov for å skape en så enkel modell som mulig. Dette forenkler de kvalitative resultatene. Løsningen blir da:

$$\phi(z) = \phi(0)e^{-Cz} \tag{7}$$

der $C = \sqrt{3\mu_a(\mu_s' + \mu_a)}$, og $\phi(0) = \frac{1}{2\delta\mu_a}$, der δ er den optiske penetrasjonsdybden. Denne er definert som dybden z der energien har blitt redusert til 36 %. Fra dette får vi et uttrykk for penetrasjonsdybden i vevet:

$$\delta = \frac{1}{C} = \frac{1}{\sqrt{3\mu_a(\mu_s' + \mu_a)}} \tag{8}$$

Dette gir oss en dybde z hvor $\phi(z) = \phi(0)e^{-1}$. Nå har vi en forståelse av hvordan fotoner oppfører seg i huden, noe som er relevant for å estimere puls ved hjelp av optiske metoder.

2.3 Reflektans og Transmittans

Reflektans og transmittans er to viktige måleparametre når det gjelder fotontransport gjennom biologiske medier. Disse begrepene hjelper til med å kvantifisere hvor mye av det innkommende lyset som henholdsvis reflekteres tilbake til kilden og transmitteres gjennom mediet.

2.3.1 Reflektans

Reflektans er definert som forholdet mellom den energimengden som reflekteres tilbake fra mediet og den opprinnelige energimengden som treffer mediet. I biologiske medier bidrar både absorpsjon og spredning til reflektansen. [2] Formelen for reflektans er gitt ved:

$$R = \frac{\phi(z = 0)}{\phi_0} \tag{9}$$

hvor ϕ_0 er den innkommende fluensraten og $\phi(z = 0)$ er fluensraten ved overflaten. Reflektansen kan brukes til å vurdere hvor mye av lyset som effektivt spres tilbake fra vevet, noe som er særlig nyttig i diagnostiske applikasjoner som bruker overflatereflekSjon for å innhente informasjon om vevsegenskaper.

2.3.2 Transmittans

Transmittans måler mengden lys som passerer helt gjennom mediet og kommer ut på den andre siden relativt til den innkommende energien. Dette er spesielt viktig i anvendelser hvor dyp penetrasjon av lys er nødvendig, som i tilfeller av dyp vevsbildebehandling eller terapeutiske applikasjoner hvor lys må nå spesifikke indre strukturer. [2] Transmittansen er definert som:

$$T = \frac{\phi(z = d)}{\phi(z = 0)} = e^{-Cd} \tag{10}$$

hvor d er tykkelsen på mediet og $C = \sqrt{3\mu_a(\mu_s' + \mu_a)}$. Transmittansen gir innsikt i hvordan lys absorberes og spres gjennom hele tykkelsen av vevet og er derfor viktig for å forstå lysens interaksjoner i mer komplekse biologiske strukturer.

2.4 Regning på Pulsutslag

Pulsutslag beskriver hvordan endringer i blodvolumet påvirker de optiske målingene. Denne modulasjonen av lysintensiteten forårsaket av hjerteslagene er grunnlaget for mange biooptiske teknikker, inkludert pulsoximetri og fotoplethysmografi.

2.4.1 Matematisk Modellering av Pulsutslag

For å kvantifisere pulsutslagene, betrakter vi hvordan endringer i blodvolum påvirker transmittansen og reflektansen av lys gjennom vevet. Når hjertet pumper blod, endres blodvolumet i arteriene, noe som fører til periodiske variasjoner i både absorpsjon og spredning av lys i vevet. [3] Denne prosessen kan modelleres ved hjelp av en matematisk tilnærming der lysintensiteten er en funksjon av blodvolumet:

$$I(t) = I_0 e^{-\mu_a(t)L} \tag{11}$$

hvor $I(t)$ er intensiteten av det mottatte lyset ved tid t, I_0 er den opprinnelige intensiteten, $\mu_a(t)$ er den tidavhengige absorpsjonskoeffisienten som varierer med blodvolumet, og L er banen lyset tar gjennom vevet.

2.4.2 Analyse av Pulsamplitude

Pulsamplituden kan deretter utledes fra de observerte variasjonene i lysintensiteten over tid. Ved å anvende Fourier-analyse eller tilsvarende signalbehandlingsmetoder, kan man isolere de periodiske komponentene som korresponderer med hjerteslagene fra andre støykomponenter i signalet.

En vanlig måte å uttrykke pulsamplitude på er ved kontrastformelen:

$$K = \frac{|I_{\max} - I_{\min}|}{I_{\min}} \tag{12}$$

hvor $I_{\max}$ og $I_{\min}$ er henholdsvis maksimal og minimal intensitet av det detekterte lyset under ett pulsslag. Kontrasten, K, gir et mål på hvor tydelig pulsutslaget er i forhold til støyen i målingene og er avgjørende for å vurdere kvaliteten på de optiske pulsmålingene.

2.5 Signalbehandling - FFT

I signalbehandlingsdelen av denne labben brukes Fast Fourier Transform (FFT) for å analysere de biologiske signalene som er samlet inn via det oppsatte systemet. FFT tillater oss å konvertere signaler fra tidsdomenet, som er registrert av Pi-kameraet, til deres representasjoner i frekvensdomenet. Dette er spesielt nyttig for å identifisere og analysere frekvenskomponentene i pulssignalene som er fanget opp, slik som å observere pulsens harmoniske forstyrrelser

eller for å vurdere signal-to-noise-ratio (SNR). Videre er det mulig å evaluere systemets evne til effektivt å filtrere ut uønskede frekvenser. [5]

2.6 Signal-to-Noise Ratio i Pulssammenheng

Signal-to-Noise Ratio (SNR) er et sentralt mål i alle former for signalanalyse og spiller en spesielt viktig rolle i analyse av biologiske signaler som pulsmålinger. SNR i pulssammenheng beskriver forholdet mellom signalets styrke (pulsamplitude) og bakgrunnsstøyen som påvirker målingen.[2]

2.6.1 Definisjon av SNR

SNR kan defineres på flere måter avhengig av konteksten og den spesifikke applikasjonen. I konteksten av puls målinger, kan SNR uttrykkes som:

$$\text{SNR} = \frac{\mu_{\text{signal}}}{\sigma_{\text{noise}}} \tag{13}$$

hvor μ_{signal} representerer gjennomsnittlig verdi av signalamplituden over flere pulssykluser, og σ_{noise} er standardavviket til målestøyen.

2.6.2 Betydning av Høy SNR

En høy SNR indikerer at signalet (pulsen) er tydelig og lett skillelig fra støy, noe som gjør det lettere å utføre nøyaktige målinger og pålitelig deteksjon av pulssignal. Dette er spesielt viktig i kliniske og mobile helseovervåkningsapplikasjoner hvor nøyaktighet er avgjørende. [6]

2.6.3 FFT-basert Tilnærming til SNR

Når nøyaktige karakteristikker av støy eller pulsens amplitude ikke er kjent, kan en FFT-basert metode anvendes for å estimere SNR, som gir en detaljert analyse av signalenes frekvenskomponenter.

Steg for FFT-basert SNR-beregning

1. Utfør en Fast Fourier Transform (FFT) på det tidsbaserte signalet for å konvertere det til frekvensdomenet.
2. Identifiser frekvensbåndet som tilsvarer pulssignalet og skill dette fra øvrige frekvensbånd som representerer støy.
3. Beregn total signalstyrke ved å integrere effekten (amplitudens kvadrat) over pulsfrekvensbåndet og total støystyrke over de resterende frekvensbåndene.

4. Estimer SNR ved følgende formel:

$$\mathrm{SNR}_{\mathrm{FFT}} = \frac{\sum \text{signalbåndeffekt}}{\sum \text{støybåndeffekt}} \tag{14}$$

Denne FFT-baserte tilnærmingen til SNR-beregning er viktig for å sikre nøyaktige og pålitelige målinger av pulssignaler, spesielt i kliniske og overvåkningsapplikasjoner der kvaliteten på biomedisinske data er avgjørende.

2.6.4 Vindusfunksjoner

Vindusfunksjoner spiller en viktig rolle i digital signalbehandling, spesielt når det gjelder spektralanalyse. En vindusfunksjon anvendes på et signal for å minimere effektene av spektral lekkasje ved å redusere signalverdiene ved endepunktene til null. [5]

Hamming Vindu

Hamming-vinduet er en populær vindusfunksjon brukt i signalbehandling for å redusere spektrallekkasje ved utføring av en Fast Fourier Transform (FFT). Det er spesielt designet for å minimere den første sideloben i vinduets frekvensrespons, som gir bedre frekvensselektivitet sammenlignet med et rektangulært vindu. [7]

Hamming-vinduet kan beskrives med følgende formel:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{15}$$

hvor n er indeksen til datapunktet i vinduet, og N er totalt antall datapunkter i vinduet. Dette vinduet reduserer amplituden på frekvenskomponentene ved endene av datavinduet, noe som resulterer i mindre spektral lekkasje og forbedret frekvensoppløsning.

Hamming-vinduet er ideelt for applikasjoner der presis frekvensanalyse er viktg, slik som i lydanalyse og vibrasjonsanalyse hvor man ønsker å isolere bestemte frekvenskomponenter med høy nøyaktighet.

2.6.5 Zero-padding i Spektralanalyse

Zero-padding er en teknikk som brukes for å øke antallet frekvenspunkter i det Fourier-transformerte spekteret. Denne metoden innebærer å legge til nuller på slutten av et signal før utførelsen av en Fourier-transformasjon. [5] Selv om zero-padding gir en finere oppløsning i frekvensdomenet ved å gi flere punkter mellom eksisterende frekvenskomponenter, endrer det ikke den faktiske frekvensoppløsningen som er definert av evnen til å skille mellom nærliggende frekvenser. Frekvensoppløsningen er fastsatt av den totale opptakstiden for signalet, og denne påvirkes ikke av zero-padding.

Zero-padding er nyttig for å gjøre spekteret enklere å tolke visuelt, men tilfører ikke ny informasjon om signalets frekvensinnhold. Det er derfor viktig å forstå denne teknikkens begrensninger og korrekt anvendelse for å unngå feiltolkninger i spektralanalysen.

2.7 Relevante labforberedelser

I vedlegg B er det en gitt kode som skal brukes til å løse relevante forberedelsesoppgaver før labben.

2.7.1 Penetrasjonsdybde i menneskevev

Det var ønskelig å regne ut penetrasjonsdybden for et typisk menneskevev for hver av bølgelengdene RGB-kameraet er sensitivt for.

Det ble funnet at RGB-kameraet er sensitivt for, $\lambda_{red} = 600nm$, $\lambda_{green} = 520nm$ og $\lambda_{blue} = 460nm$. Disse verdiene ble satt inn i koden i vedlegg B.

```
# find penetration depth for each wavelength
penetrations = np.sqrt(1/(3*mua*(musr + mua)))
print(penetrations)

OUTPUT: [0.0015696  0.00083338 0.00059616]
```

Kode 1: Utregning av Penetrasjonsdybde.

OUTPUT viser penetrasjonsdybden i [ENHET!] ved kritisk bølgelengde for henholdsvis rød, grønn og blå fargekanal.

2.7.2 Prosentandel av lys gjennom finger

Det var også ønskelig å finne prosentandelen av lys som kommer ut på andre siden av fingeren ved RGB-kameraet sine sensitive bølgelengder.

```
# finger thickness
finger_thickness = 0.01 # meters
# calculate the percentage of light that comes out the other side for each
    wavelength
percentages = np.exp(-np.sqrt(3*mua*(musr + mua))*finger_thickness)
print(percentages)

OUTPUT: [1.71037241e-03 6.14862493e-06 5.18974749e-08]
```

Kode 2: Utregning av Lysprosent.

OUTPUT viser prosentandelen av lys som kommer gjennom på andre siden av en finger ved kritisk bølgelengde for henholdsvis rød, grønn og blå fargekanal.

2.7.3 Reflektans i menneskevev

Det er også hensiktsmessig å undersøke reflektansen til menneskevev. Dette ble gjort med kode 3.

```
# find reflectance depth for each wavelength
reflectances = np.sqrt((3*musr/mua+1))
print(reflectances)

OUTPUT: [12.64901168  8.64389252  8.05833808]
```

Kode 3: Utregning av Reflektansdybde

OUTPUT viser reflektansen ved kritisk bølgelengde for henholdsvis rød, grønn og blå fargekanal.

2.7.4 Kontrast

Det er nyttig å se på kontrast ettersom det gir et tydelig mål på kvaliteten av lesningnen på pulssignalet.

```
import numpy as np

# Given data
muabo = np.genfromtxt("muabo.txt", delimiter=",")
muabd = np.genfromtxt("muabd.txt", delimiter=",")
wavelength = np.array([600, 520, 460])
bvf_blood_vessel = 1.0  # Blood volume fraction for the blood vessel (100%)
bvf_tissue = 0.01  # Blood volume fraction for the tissue (1%)
oxy = 0.8  # Blood oxygenation
mua_other = 25  # Background absorption
finger_thickness = 0.0003  # Thickness of the blood vessel in meters

# Functions to calculate absorption coefficient for oxy and deoxy blood
def mua_blood_oxy(x): return np.interp(x, muabo[:, 0], muabo[:, 1])
def mua_blood_deoxy(x): return np.interp(x, muabd[:, 0], muabd[:, 1])

# Calculate absorption coefficients for blood vessel and tissue
mua_blood_vessel = (mua_blood_oxy(wavelength)*oxy + mua_blood_deoxy(wavelength)*(1-oxy)) * bvf_blood_vessel + mua_other
mua_tissue = (mua_blood_oxy(wavelength)*oxy + mua_blood_deoxy(wavelength)*(1-oxy)) * bvf_tissue + mua_other

# Reduced scattering coefficient
musr = 100 * (17.6*(wavelength/500)**-4 + 18.78*(wavelength/500)**-0.22)

# Calculate percentage of light that comes out the other side for blood vessel and tissue
percentages_blood_vessel = np.exp(-np.sqrt(3*mua_blood_vessel*(musr + mua_blood_vessel))*finger_thickness)
percentages_tissue = np.exp(-np.sqrt(3*mua_tissue*(musr + mua_tissue))*finger_thickness)

# Calculate contrast
contrast = np.abs((percentages_blood_vessel - percentages_tissue)/percentages_tissue)

print(percentages_blood_vessel, percentages_tissue, contrast)

```

```
33 OUTPUT: [1.52293665e-01 1.29423829e-03 2.92101402e-05] [0.8260241  0.69769135
       0.60458085] [0.81563048 0.99814497 0.99995169]
```

Kode 4: Utregning av Kontrast

OUTPUT viser prosentandel av lys som kommer gjennom boldårene, prosentandel av lys som kommer gjennom vev og kontrasten mellom de ved de kritisk bølgelengdene for henholdsvis rød, grønn og blå fargekanal.

2.7.5 Egnethet til pulsmåling

De ulike fargekanalene har ulik egnethet til pulsmåling. Vi forventer at grønn er best egnet. Blått har den beste kontrasten, men grønn er ikke mye dårligere. Blått lys vil derimot være vanskelig å måle ettersom mindre blått lys kommer gjennom fingeren. Derfor tror vi grønn fargekanel vil fungere best.

3 Systemoppsett

I denne seksjonen presenteres oppsettet som ble benyttet for å gjennomføre de optiske pulsmålingene. Oppsettet inkluderer både det fysiske oppsettet og den programvarekonfigurasjonen som ble brukt for signalbehandling. Etter å ha verifisert systemet med fem basis målinger forklart under, gjennomførte vi to robusthetstester. Først ved å se på forskjellen av å måle pulsen på en kald og varm finger etterfulgt av måling av puls gjennom en øreflipp.

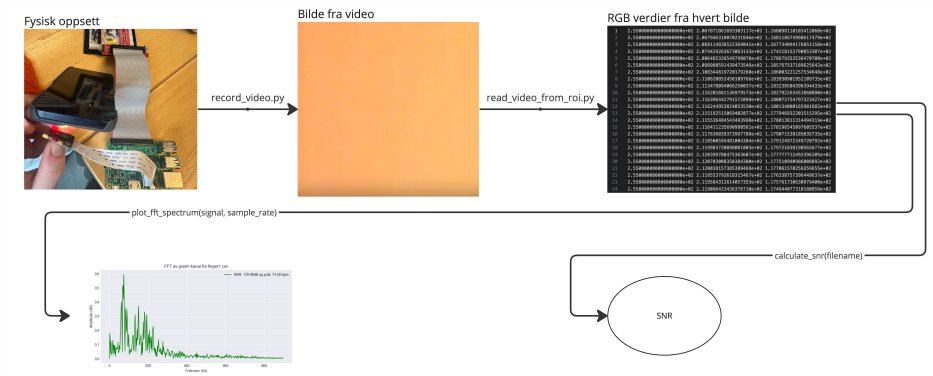
3.1 Utstyrsliste

Følgende hardvare ble brukt under labben:

- Raspberry Pi 4 Model B.
- Raspberry Pi Kamera Modul V2.
- Iphone 13 lykt.

3.2 Fysisk oppsett

Et enkelt flytdiagram er vist i figuren 3 under der koden brukt i *record_video.py* og *read_video_from_roi.py* kan sees i vedlegg *D* og funksjonene *calculate_snr*(filename) og *plot_fft_spectrum*(signal, sample rate) kan sees i vedlegg *A*.



Figur 3: Flytdiagram av oppsettet fra fysisk oppsett til signalbehandling

Figur 3 viser hvordan vi gjennomførte de fem basismålingene med unntak av en lysskjerm som ble holdt over systemet for å forhindre forstyrrelser fra omgivelsene, noe som kan gi mer støy og dårligere resultat. Det samme systemet ble brukt for å gjøre begge robusthetstestene med kald og varm finger, og øreflipp-testen. Den utdelte koden i vedlegg D tar et 30 sekunders video opptak som vi kan ekstrahere endringene fargedataen fra hvert bilde i videoen.

3.3 Signalbehandling

Programvaren som ble skrevet i Python, ble hovedsaklig brukt for å utføre nødvendig signalbehandling på dataen som ble målt. Som nevnt over, er koden for å kjøre opptak av video og overføring av video til RGB verdier i CSV-fil, utdelt og ligger i vedlegg D.

Signalbehandlingen som ble gjennomført på målingene kan deles opp i følgende seksjoner, der vi først gjennomførte FFT på målingene for å finne den mest tydelige frekvensen før vi også regnet ut SNR på dataen vi målte. Som beskrevet i 2 ser vi kun på grønn fargekanal.

3.3.1 FFT

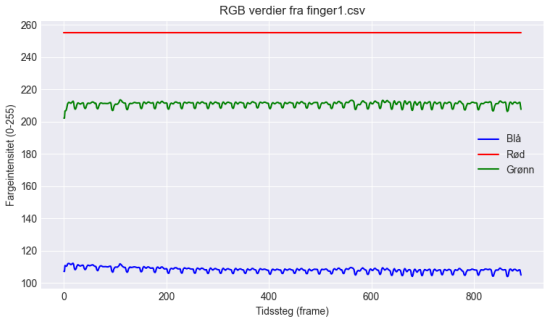
Fra 3 ser vi at FFT funksjonen *plot_fft_spectrum*(signal, sample rate), vedlagt i vedlegg A, henter ut verdiene fra en bestemt fargekanal for å generere en plot med frekvensspektrumet. Ved å se på frekvensen f_{peak} med sterkest amplitude i frekvensspektrumet, kan vi finne målt hjerteslag per minutt (bpm) ved å multiplisere frekvensen f_{peak} med 60 sekunder.

3.3.2 SNR

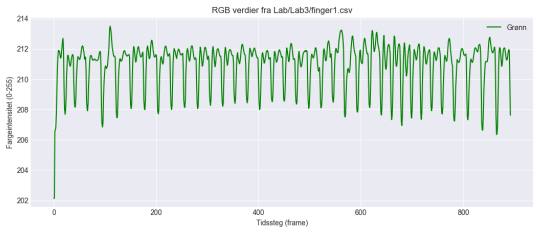
Funskjonen *calculate_snr*(filename) for å regne ut SNR på målt data, vedlagt i vedlegg D, behandles den grønne fargekanalen ved først å normalisere signalet. Normaliseringen gjøres ved å trekke fra gjennomsnittsverdien fra hver datapunkt for sentrere dataene rundt null. Etter normalisering utføres en Fast Fourier Transform (FFT) for å transformere det tidsbaserte signalet til sitt frekvensdomene, hvor den dominerende frekvensen kan identifiseres tydelig. SNR beregnes ved å dele amplitudeverdien av denne dominerende frekvensen, funnet i frekvensspekteret, med standardavviket til det opprinnelige, normaliserte signal. Denne metoden gir et mål på hvor distinkt puls-signalet er i forhold til bakgrunnsstøyen, og brukes for å vurdere kvaliteten på signalene som er oppfanget.

4 Resultater og diskusjon

Figur 4a viser hvordan fargeverdiene endret seg i en av de fem basismålinger som ble gjennomført. Her ser vi at grønn fargekanal har høyere intensitet enn blå kanal og at rød fargekanal er helt saturert. Dette stemmer godt med teorien og grønn kanal, vist i figur 4b viser en tydelig periodisk variasjon, og ble derfor brukt videre i signalbehandlingen.



(a) RGB verdier.

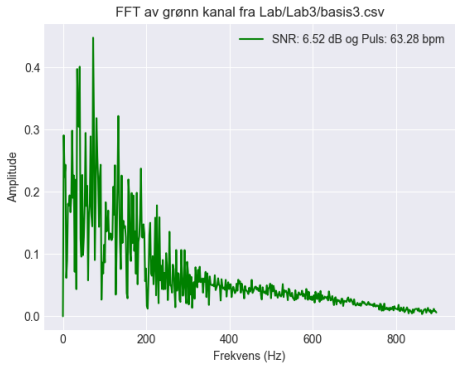


(b) Fokus på grønn fargekanal.

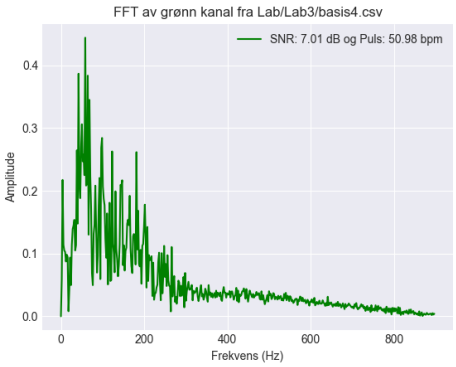
Figur 4: Rådataen til fargekanalene fra en av basismålingene.

Figur 5 viser plot av frekvensspektrumet til fure basismålinger, der man kan hente ut frekvensen med størst amplitude for å estimere gjennomsnitts pulssignal. 1 viser resultatene fra alle fem basismålingene inkludert robusthetstesten beskrevet lengre ned.

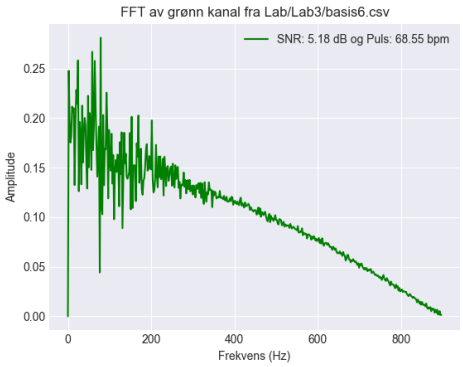
Alle målingene som ble gjennomført ble verifisert ved hjelp av en pulsklokke av typen Garmin Fenix 6. Bruk og implementasjon av Zero-padding og vindu-funksjoner under SNR beregningene for å forbedre estimatene kan leses mer i rapporten om Systemoppsett [5].



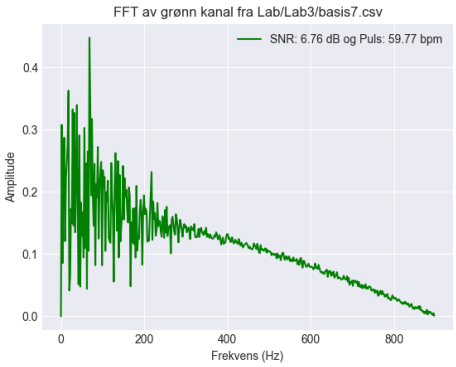
(a) Frekvensspektrum fra basismåling 1.



(b) Frekvensspektrum fra basismåling 2.



(c) Frekvensspektrum fra basismåling 3.



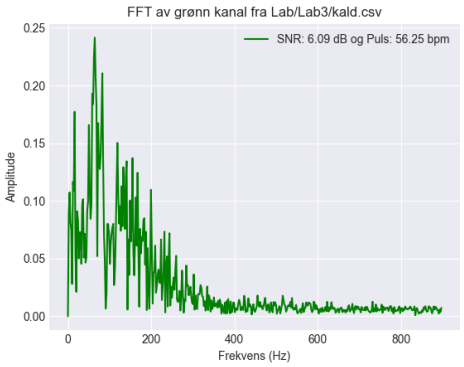
(d) Frekvensspektrum fra basismåling 4.

Figur 5: Sammenlikning av frekvensspektrummet til fire av fem basismålinger.

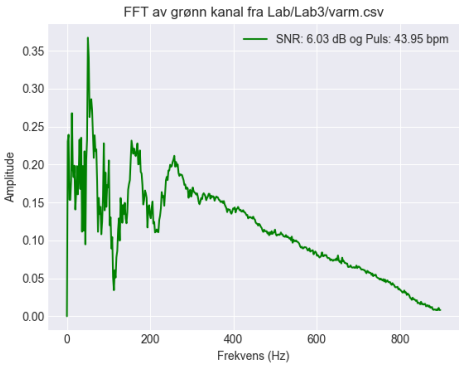
Tabell 1: SNR og estimert gjennomsnittspuls.

Måling	**SNR (dB)**	**Estimert puls (bpm)**	**Observert puls fra pulsklokke (bpm)**
Basismåling 1	6.52	63.3	60
Basismåling 2	7.01	51	55
Basismåling 3	5.18	68.6	63
Basismåling 4	6.78	59.8	60
Basismåling 5	7.44	60.0	59
Gjennomsnitt	**6.59**	**60.54**	**59.4**
Standardavvik	**0.77**	**5.73**	**2.58**
Varm finger	6.09	58.3	61
Kald finger	6.03	44.0	49

For å videre undersøke robustheten til systemet ble det gjennomført en test der estimert gjennomsnittspuls med oppvarmet og nedkjølt finger ble målt. Figur 6 viser sammenlikningen av frekvensspektrumet til disse målingene. Økt temperatur i fingrene øker også blodtilførsel, noe som gir et tydeligere signal og kraftigere amplitude i figur 6b.



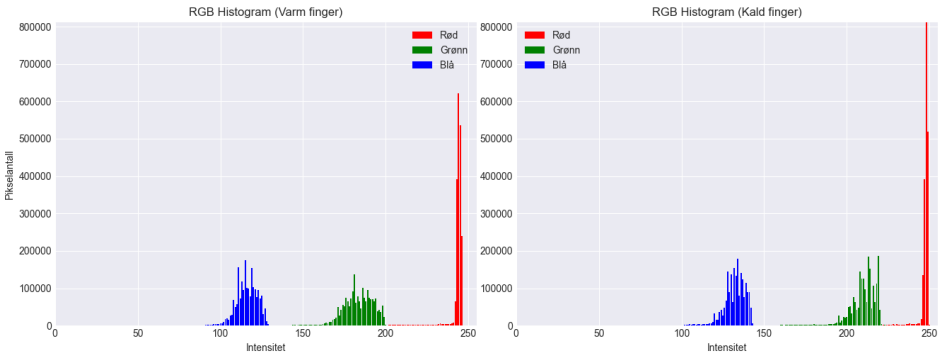
(a) Frekvensspektrum for måling av nedkjølet finger.



(b) Frekvensspektrum for måling av oppvarmet finger.

Figur 6: Sammenlikning av frekvensspektrum mellom kald og varm finger.

Det kan også være interessant å se på RGB histogrammet til de to målingene vist i 7, der det er tydelig mer intensitet i alle fargekanalene fra varm finger.



Figur 7: RGB histogram for kald og varm finger.

5 Konklusjon

I denne rapporten er det beskrevet en gjennomgang av optiske målemetoder og de teoretiske grunnlagene som er nødvendige for å utføre pulsregistrering ved hjelp av optisk sensorikk. Vi har brukt et målesystem som bruker en Raspberry Pi, med tilhørende kamera til å fange opp blodvolumendringer i huden, som reflekteres i pulssignalene. Dataene ble samlet inn og behandlet for å fjerne støy og forbedre signalens klarhet, blant annet ved hjelp av Fast Fourier Transform og filtreringsteknikker som bruk av Hamming-vindu og zero-padding.

Referanser

[1] Linköping University: *Biomedical Optics*. https://liu.se/en/research/biomedical-optics, April 2024. Accessed: 2024-04-25.

[2] NTNU-fagstab: *Del IV Optikk*, Februar 2024. Accessed: 2024-04-25.

[3] Bigio, Irving J. og Sergio Fantini: *Quantitative Biomedical Optics: Theory, Methods, and Applications*. Cambridge Texts in Biomedical Engineering. Cambridge University Press, 2016, ISBN 9780521876568. https://books.google.no/books?id=X7AkCwAAQBAJ.

[4] Svaasand, Lars O, Thorsten Spott, Joshua B Fishkin, Tuan Pham, Bruce J Tromberg og Michael W Berns: *Reflectance measurements of layered media with diffuse photon-density waves: a potential tool for evaluating deep burns and subcutaneous lesions*. Physics in Medicine Biology, 44(3):801, mar 1999. https://dx.doi.org/10.1088/0031-9155/44/3/020.

[5] Fløan, Freider og Theodor Qvist: *Signalbehandling og spektralanalyse av måleplattform*. Labrapport, Norwegian University of Science and Technology (NTNU), April 2024. Gruppe 44, TTT4280 - Sensor og instrumentering.

[6] Sheldon, Robert: *Signal-to-Noise Ratio (SNR)*. https://www.techtarget.com/searchnetworking/definition/signal-to-noise-ratio, 2021. Last updated: August 2021, Accessed: 2024-04-25.

[7] Harris, Frederic J.: *Hamming Window - an overview*. https://www.sciencedirect.com/topics/engineering/hamming-window. Accessed: 2024-04-25.

A Vedlegg A

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.fft import fft, fftfreq
plt.style.use('seaborn-darkgrid')

def plot_rgb_values(filename):
    data = pd.read_csv(f'{filename}', delimiter=" ", header=None, names=['Red', 'Green', 'Blue'])
    plt.plot(data.index, data['Blue'], color='blue', label='B1  ')
    plt.plot(data.index, data['Red'], color='red', label='R d ')
    plt.plot(data.index, data['Green'], color='green', label='Gr nn ')
    plt.xlabel('Tidssteg (frame)')
    plt.ylabel('Fargeintensitet (0-255)')
    plt.title(f'RGB verdier fra {filename}')
    plt.legend()
    plt.show()

def calculate_snr(filename):

    data = pd.read_csv(f'{filename}', delimiter=" ", header=None, names=['Red', 'Green', 'Blue'])
    signal = data['Green'] - data['Green'].mean()

    N = signal.size
    T = 1.0 / 30

    yf = np.fft.fft(signal)
    xf = np.fft.fftfreq(N, T)[:N//2]

    amplitudes = np.abs(yf)
    max_amp_index = np.argmax(amplitudes)

    highest_freq = xf[max_amp_index]
    highest_amp = amplitudes[max_amp_index]

    snr = highest_amp / data['Green'].std()

    return snr

def plot_fft_spectrum(signal, sample_rate):
    N = len(signal)
    T = 1.0 / sample_rate
    yf = np.fft.fft(signal)
    xf = np.fft.fftfreq(N, T)[:N//2]
    amplitudes = np.abs(yf)
    max_amp_index = np.argmax(amplitudes)
    highest_freq = xf[max_amp_index]
    highest_amp = amplitudes[max_amp_index]
    snr = calculate_snr(signal)
    # Plot the FFT spectrum
    plt.plot(xf*60, 2.0/N * np.abs(yf[0:N//2]), color='green', label=f'SNR: {snr:.2f}dB og puls: {highest_freq*60:.2f} bpm')
    plt.xlabel('Frekvens (Hz)')
    plt.ylabel('Amplitude (dB)')
```

```
53      plt.title('FFT av gr nn kanal')
54      plt.legend()
55      plt.show()
```

Kode 5: Funksjon for å regne ut SNR og plotte FFT.

B Vedlegg B

```
import numpy as np

muabo = np.genfromtxt("./muabo.txt", delimiter=",")
muabd = np.genfromtxt("./muabd.txt", delimiter=",")

red_wavelength = None # Replace with wavelength in nanometres
green_wavelength = None # Replace with wavelength in nanometres
blue_wavelength = None # Replace with wavelength in nanometres

wavelength = np.array([red_wavelength, green_wavelength, blue_wavelength])

def mua_blood_oxy(x): return np.interp(x, muabo[:, 0], muabo[:, 1])
def mua_blood_deoxy(x): return np.interp(x, muabd[:, 0], muabd[:, 1])

bvf = 0.01 # Blood volume fraction, average blood amount in tissue
oxy = 0.8 # Blood oxygenation

# Absorption coefficient ($\mu_a$ in lab text)
# Units: 1/m
mua_other = 25 # Background absorption due to collagen, et cetera
mua_blood = (mua_blood_oxy(wavelength)*oxy # Absorption due to
             + mua_blood_deoxy(wavelength)*(1-oxy)) # pure blood
mua = mua_blood*bvf + mua_other

# reduced scattering coefficient ($\mu_s^\prime$ in lab text)
# the numerical constants are thanks to N. Bashkatov, E. A. Genina and
# V. V. Tuchin. Optical properties of skin, subcutaneous and muscle
# tissues: A review. In: J. Innov. Opt. Health Sci., 4(1):9-38, 2011.
# Units: 1/m
musr = 100 * (17.6*(wavelength/500)**-4 + 18.78*(wavelength/500)**-0.22)

# mua and musr are now available as shape (3,) arrays
# Red, green and blue correspond to indexes 0, 1 and 2, respectively

# TODO calculate penetration depth
```

Kode 6: Utdelt kode til forberedelses oppgavene.

C Vedlegg C

```python
import matplotlib.pyplot as plt
from PIL import Image

def plot_combined_rgb_histograms(image_path1, image_path2):

    img1 = Image.open(image_path1).convert('RGB')
    img2 = Image.open(image_path2).convert('RGB')

    r1, g1, b1 = img1.split()
    r2, g2, b2 = img2.split()

    r_hist1 = r1.histogram()
    g_hist1 = g1.histogram()
    b_hist1 = b1.histogram()
    r_hist2 = r2.histogram()
    g_hist2 = g2.histogram()
    b_hist2 = b2.histogram()

    max_hist1 = max(r_hist1 + g_hist1 + b_hist1)
    max_hist2 = max(r_hist2 + g_hist2 + b_hist2)
    max_y = max(max_hist1, max_hist2)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

    ax1.bar(range(256), r_hist1, color='red', label='R d ')
    ax1.bar(range(256), g_hist1, color='green', label='Gr nn ')
    ax1.bar(range(256), b_hist1, color='blue', label='Bl  ')
    ax1.set_title('RGB Histogram (Varm finger)')
    ax1.set_xlabel('Intensitet')
    ax1.set_ylabel('Pikselantall')
    ax1.legend()
    ax1.set_xlim([0, 255])
    ax1.set_ylim([0, max_y])

    ax2.bar(range(256), r_hist2, color='red', label='R d ')
    ax2.bar(range(256), g_hist2, color='green', label='Gr nn ')
    ax2.bar(range(256), b_hist2, color='blue', label='Bl  ')
    ax2.set_title('RGB Histogram (Kald finger)')
    ax2.set_xlabel('Intensitet')
    ax2.legend()
    ax2.set_xlim([0, 255])
    ax2.set_ylim([0, max_y])

    plt.tight_layout()
    plt.show()
```

Kode 7: Funksjon for å plotte RGB histogram.

D Vedlegg D

```
1
2  """
3  Script for recording a video using the PiCam library,
4  for later post-processing and extraction of pulse signals
5  for the optics lab in the TTT4280 course.
6
7  Run without arguments to see the usage instructions.
8
9  This script was mainly written by Jon Magnus Momrak Haug and Martin Ervik in
10 preparation for the optics lab in TTT4280 in 2017.
11
12 Small modifications made by Asgeir Bjorgan in 2018 (add CLI arguments, change
13 MP4Box arguments to overwrite the container instead of adding to it, avoiding
14 the need for removing the previous output file).
15 """
16
17 import subprocess
18 import os
19 from picamera import PiCamera
20 from time import sleep
21 import sys
22
23 # Print usage instructions
24 if len(sys.argv) < 2:
25         print('Usage: python ' + sys.argv[0] + ' [path to filename]')
26         print('')
27         example_path = 'some/path/to/a/folder/optics_labs/videos/'
28         example_filename = 'finger_kurt-leif_transillumination'
29         print('Example: `python ' + sys.argv[0] + ' ' + example_path +
       example_filename + '`')
30         print('(The example will result in two files, ' + example_filename +
       '.h264 and ' + example_filename + '.mp4)')
31         exit()
32
33 # Split the root from the extension, ensure correct output file extension
34 DEFAULT_FILE_EXTENSION = '.h264'
35 root, extension = os.path.splitext(sys.argv[1])
36 if extension != DEFAULT_FILE_EXTENSION:
37         extension = DEFAULT_FILE_EXTENSION
38 h264_filename = root + extension
39
40 # PiCamera instance. This is an object which has several attributes that can
       be changed.
41 camera = PiCamera()
42
43 # The resolution can be changed, but the frame rate has to be supported by the
44 # current resolution.
45 camera.resolution = (1640, 922)
46 camera.framerate = 40
47
48 # Set a low ISO. Adjusts the sensitivity of the camera. 0 means auto. 10 most
       probably is the same
49 # as 100, as picamera doc says it can only be set to 100, 200, 300, ..., 800.
50 # Might probably want to set this to lowest possible value not equal to 0, but
        can also be tuned.
```

```
camera.iso = 10

# Add a bit of delay here so that the camera has time to adjust its settings.
# Skipping this will set a very low exposure and yield black frames, since the
# camera needs to adjust the exposure to a high level according to current
# light settings before turning off exposure compensation.
print('Waiting for settings to adjust')
sleep(2)

# switch these two off so that we can manually control the awb_gains
camera.exposure_mode = 'off'
camera.awb_mode = 'off'

# Set gain for red and blue channel.  Setting a single number (i.e.
# camera.awb_gains = 1) sets the same gain for all channels, but 1 seems to be
# too low for blue channel (frames constant black).  2 sometimes is not enough,
# and is related to what happens during the sleep(2) above.  Probably has to be
# tuned?
camera.awb_gains = (1, 2)

# Other parameters that could be tuned:
#
# * camera.exposure_compensation (should have been set to a sensible value while camera was sleeping? Might want to fix this for reproducable results)
# * camera.brightness
#
# See https://picamera.readthedocs.io/en/release-1.10/api_camera.html for description of properties,
# and inspiration for other properties to adjust.

# how long we want to record
recordTime = 30

# If we were not running the Pi headless, starting the preview would show us
# what the camera was capturing.  Now, since we run it headless, it allows the
# rest of the settings to be set.
print("Waiting for white balance to adjust")
camera.start_preview()
sleep(5)
print("Start recording to " + h264_filename)
camera.start_recording(h264_filename)

# Record for the amount of time we want
camera.wait_recording(recordTime)

# When finished, stop recording and stop preview.
camera.stop_recording()

print("Recording finished")
camera.stop_preview()

################################################################################
## The following section is where we wrap the .h264 into a mp4 container ##
################################################################################
```

```
103
104 # .h264 doesn't containing any information about the framerate or duration of
        the movie.
105 #
106 # We therefore have to wrap a container around the movie and supply this
        information manually.
107 # This can be done using the application MP4Box, e.g. `MP4Box -new -fps [fps]
        -add [filename.h264] [output_filename.mp4].
108 # We do this automatically by executing MP4Box externally from the Python
        script (it is not good practice to execute
109 # external programs like this, but we'll do it anyway.)
110 #
111 # Overview over CLI flags:
112 # * -fps: Set frames per second as according to the recorded fps
113 # * -new: Force overwrite of .mp4-file (instead of adding clip to it)
114 # * -add: Add clip in .h264-file to .mp4-file
115 print("Pack video in MP4 container")
116 mp4_filename = root + ".mp4"
117 subprocess.check_output(["ffmpeg", "-framerate", str(camera.framerate), "-i",
        h264_filename, "-c", "copy", mp4_filename])
118 print("Files saved to: " + h264_filename + " and " + mp4_filename)
```

Kode 8: Utdelt kode for opptak av video.

```
1  """
2  Read the input video file, display first frame and ask the user to select a
3  region of interest.  Will then calculate the mean of each frame within the ROI
       ,
4  and return the means of each frame, for each color channel, which is written
       to
5  file.
6
7  Similar to read_video_and_extract_roi.m, but for Python.
8
9  Requirements:
10
11 * Probably ffmpeg. Install it through your package manager.
12 * numpy
13 * OpenCV python package.
14     For some reason, the default packages in Debian/Raspian (python-opencv and
15     python3-opencv) seem to miss some important features (selectROI not
       present
16     in python3 package, python2.7 package not compiled with FFMPEG support),
       so
17     install them (locally for your user) using pip:
18     - pip install opencv-python
19     - (or pip3 install opencv-python)
20 """
21 import sys
22 import cv2
23 import numpy as np
24
25 #CLI options
26 if len(sys.argv) < 3:
27     print("Select smaller ROI of a video file, and save the mean of each image
        channel to file, one column per color channel (R, G, B), each row
       corresponding to a video frame number.")
28     print("")
```

```
29      print("Usage:\npython " + sys.argv[0] + " [path to input video file] [path
        to output data file]")
30      exit()
31  filename = sys.argv[1]
32  output_filename = sys.argv[2]
33
34  #read video file
35  cap = cv2.VideoCapture(filename, cv2.CAP_FFMPEG)
36  if not cap.isOpened():
37      print("Could not open input file. Wrong filename, or your OpenCV package
        might not be built with FFMPEG support. See docstring of this Python
        script.")
38      exit()
39
40  num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
41  fps = cap.get(cv2.CAP_PROP_FPS)
42
43  mean_signal = np.zeros((num_frames, 3))
44
45  #loop through the video
46  count = 0
47  while cap.isOpened():
48      ret, frame = cap.read() #'frame' is a normal numpy array of dimensions [
        height, width, 3], in order BGR
49      if not ret:
50          break
51
52      #display window for selection of ROI
53      if count == 0:
54          window_text = 'Select ROI by dragging the mouse, and press SPACE or
        ENTER once satisfied.'
55          ROI = cv2.selectROI(window_text, frame) #ROI contains: [x, y, w, h]
        for selected rectangle
56          cv2.destroyWindow(window_text)
57          print("Looping through video.")
58
59      #calculate mean
60      cropped_frame = frame[ROI[1]:ROI[1] + ROI[3], ROI[0]:ROI[0] + ROI[2], :]
61      mean_signal[count, :] = np.mean(cropped_frame, axis=(0,1))
62      count = count + 1
63
64  cap.release()
65
66  #save to file in order R, G, B.
67  np.savetxt(output_filename, np.flip(mean_signal, 1))
68  print("Data saved to '" + output_filename + "', fps = " + str(fps) + " frames/
        second")
```

Kode 9: Utdelt kode for konvertering av fargeverdiene til CSV fil.