Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: **Automatisering og elektronikkdesign**	Vårsemesteret, 2021 Åpen
Forfattere: **Christian Kvalvåg**	
Fagansvarlig: **Tormod Drengstig** Veileder(e): **Tormod Drengstig**	
Tittel på bacheloroppgaven: **Bildebehandling og kommunikasjonmellom Raspberry PI og PLS** Engelsk tittel: **Image processing and communication between Raspberry PI and PLC**	
Studiepoeng: **20**	
Emneord: **PLS, Python, Raspery PI, USB kamera, IDS kamera**	Sidetall: 30 +vedlegg/annet: 20 Stavanger, 24.07.2021 dato/år

Forord

Jeg ønsker å takke Norweigan Lobster Farm for oppgaven de utlyste. Oppgaven har bydd på flere utfordringer enn ventet, og har vært veldig tidkrevende, og lærerik. Underveis i oppgaven har jeg byttet komponenter og endret tankemåte for å løse oppgaven ved flere anledninger. Da jeg fikk tildelt oppgaven, gikk det noen uker før jeg fikk kommet skikkelig i gang på grunn av venting på deler. Denne tiden brukte jeg på å planlegge så godt jeg kunne. For å være så effektive som mulig, leste jeg meg opp på problem- stillingene, og løste hver del for seg. I oppgaven var det nødvendig å finne ut av hvilken kamera er som skulle brukes. Det var nødvendig med en del oppkobling og lodding for å få komponentene til å kommunisere sammen. Denne delen av oppgaven har gått veldig greit, da begge deltakerne i oppgaven har praktisk erfaring som automatikere. Jeg ønsker å rette en stor takk til Tormod Drengstig ved IDE på Universitetet i Stavanger for god veiledning, oppfølging, tips og råd om oppgaveskriving i en krevende tid pga. koronaviruset som har herjet. Dette har heldigvis ikke bydd på veldig store problemer for meg, da jeg har fått lov å låne alt utstyret nødvendig for oppgaven med hjem. Jeg ønsker også å rette en takk til Simen Kvalvåg, min bror, for veiledning og diskusjons partner rundt Python kode. Tillegg ønsker jeg å rette en takk til Elin Kvalvåg, min mor, for korrektur lesning.

Forfatter av rapporten:

Christian Kvalvåg er student ved UiS og går studieretningen Y-vei automatisering og elektronikkdesign. Forfatter har yrkesfaglig bakgrunn innen elektriker, med fagbrev.

Innhold

Figurer

Tabeller

Listings

1 Innledning

På Finnøy utenfor Stavanger ligger et verdensledende selskap innen landbasert oppdrett av hummer, Norwegian Lobster Farm. Ved hjelp av resirkulering av sjøvann (RAS), robotisering og bildebehandling ønsker Norwegian Lobster Farm å muliggjøre storskala, bærekraftig, hummerproduksjon [6]. Selve konseptet bygger på et oppdrettsanlegg som bruker nyere teknologi til å røkte, fôre, overvåke og høste hummere i egne bur [7]. Mellom 2016 og 2018 var selskapet en del av EU-prosjektet DEVAELA", et prosjekt som søkte å utvikle og demonstrere et system fra daglig fôring, overvåkning og håndtering av europeisk hummer gjennom bruk av single-cageteknologi [5][3]. I 2019 ble de innlemmet i EU's innovasjonsprosjekt 'Horizon 2020' for (prosjektnavn: 'Automarus'). Målet er å utvikle det første fullautomatiserte storskala oppdrettet for europeisk hummer ved bruk av resirkulert sjøvann (RAS) [4].

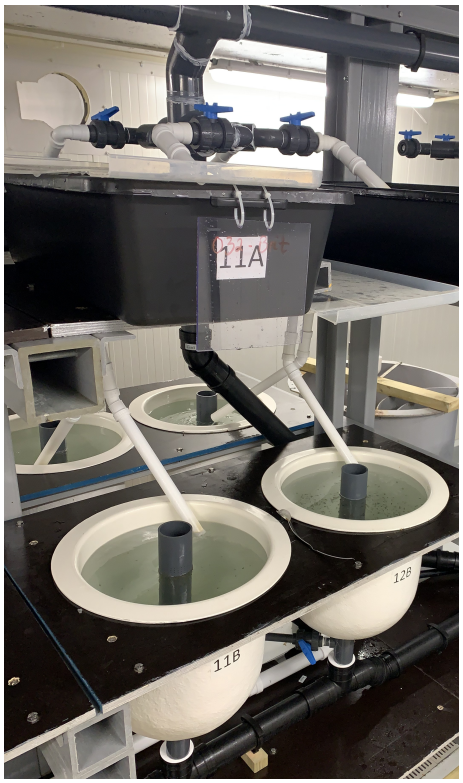
Som en del av Automarus samarbeider Norwegian Lobster Farm med forsknings- og utdanningsinstitusjonen Universitet i Stavanger. Følgende har det blitt utarbeidet bacheloroppgaver innen studieretningen "Automatisering og Elektronikkdesign", hvor denne oppgaven skal utvikle et system for å overvåke hummerene ved hjelp av bildebehandling. Basert på et Python-script som kommuniserer med en (PLS) tar systemet daglige bilder av hummerene i hvert bur for å undersøke om hummeren er i live eller om den er dø. Dette gjøres ved å bruke bildegjenkjenningsteknologi for å sammenligne hummerens posisjon dag for dag. Ved å analysere hummerens posisjonsdata kan systemet automatisk finne ut om hummerene lever. Systemet vil i tillegg laste opp og lagre alle bilder i en sky-basert database slik at bildene kan brukes ved senere behov.

Kapittel 2 legger grunnlaget for oppgaven ved å gi en generell introduksjon til hummeroppdrett og teknologien/løsningen til Norwegian Lobster Farm. Videre gis det en beskrivelse av komponentene som er relevant for oppgaven i kapittel 3, før kapittel 4 gir en detaljert beskrivelse av hvordan systemet som utvikles fungerer. I kapittel 5 presenteres resultatene av oppgaven, før den avsluttes med en diskusjon og konklusjon i kapittel 6.

2 Norwegian Lobster Farm - Bakgrunn og teknologi

2.1 Hummerensforløp

Norwegian Lobster Farm har utviklet metoder og biologiske protokoller som gjør det mulig å påvirke tidspunktene for gyting og klekking. Dermed kan de produsere hummeryngel med gode vekstegenskaper hele året. Biologiske protokoller med standardiserte metoder for å tidfeste modningsperioden for hummerrogn, samt analysekart av rogn for å estimere klekketidspunktet for egg er utviklet.

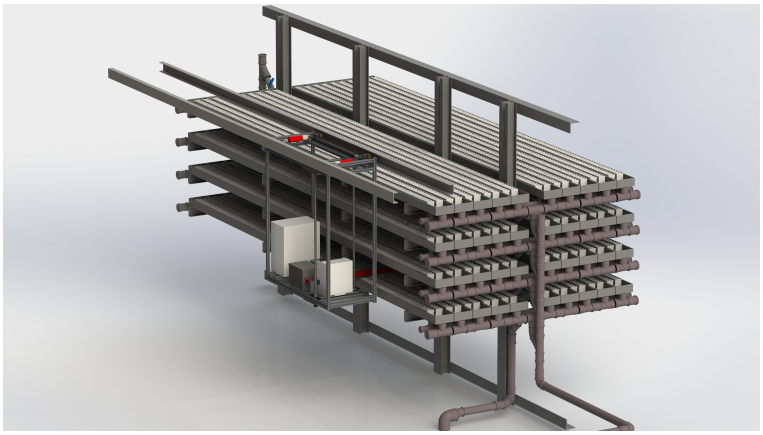


Figur 1: Anlegg for modningsperiode av små-hummer

Figur 2: Bilde av små hummer

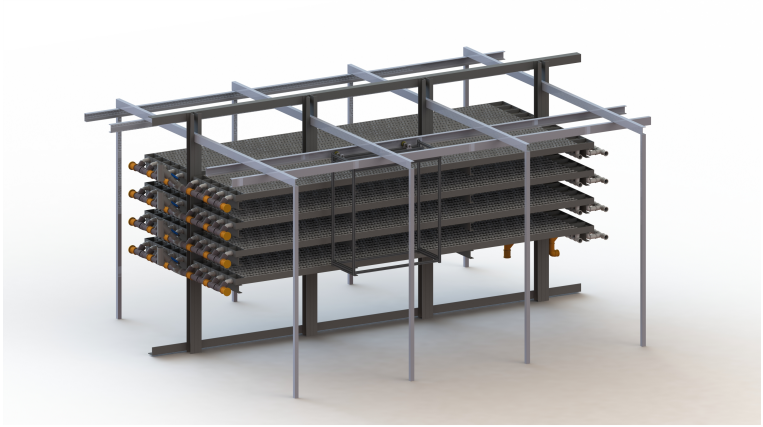
2.2 Konstruksjonen av oppdrettsanlegget

Konstruksjonen består av 4 stående bjelker som fordeler belastningen mot takbjelken og gulvbjelken. Disse 4 konstruksjon bjelkene tar vekten fra hyllene. Det er plassert 4 hyller på hver side.



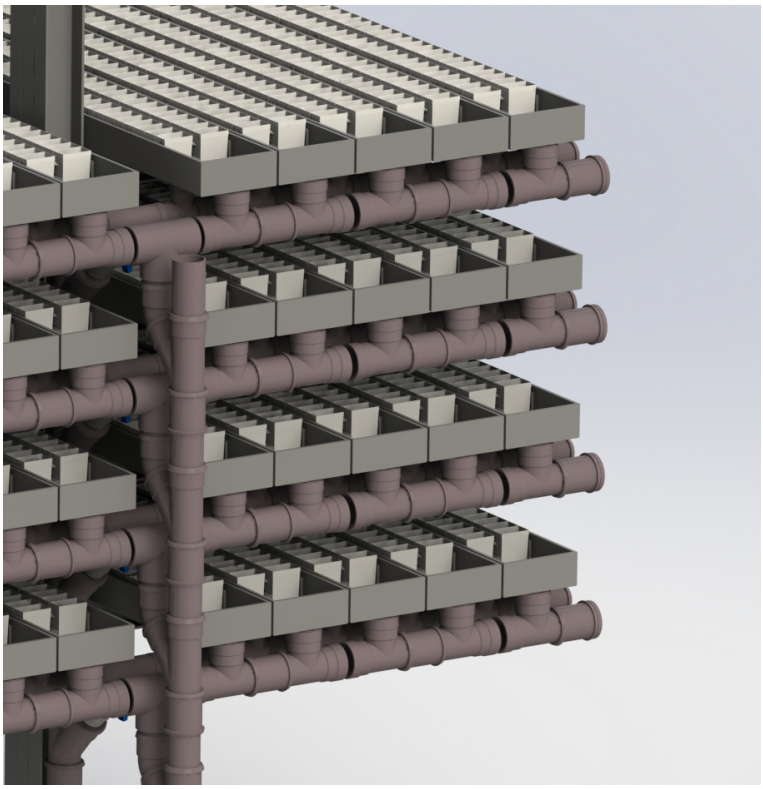
Figur 3: Framvisning av riggen

Utenfor hyllene er det montert opp 2 skinner på hver sin side. Dette kan du se i figur 4. Dette er for å kjøre en foringsvogn. Oppbygging av denne styrevognen kommer i 2.3 Styring. Denne foringsvognen er en arm som stikker ut over burene.



Figur 4: Oppbygging av strukturen på riggen

Anlegget er av en konstruksjon som gir mulighet til å fore 10 stk bur av gangen. Dette skjer igjennom vognen på siden se delkapittel 2.3. Det er total 120 rader med bur, som gir en total på 1200 bur per hylle. Konstruksjonen er bygget på den måten at det er 4 hyller på hver side. Med er det en total antall på 4800 bur per side. da på begge sidene vil det være mulighet for 9600 bur på denne riggen.

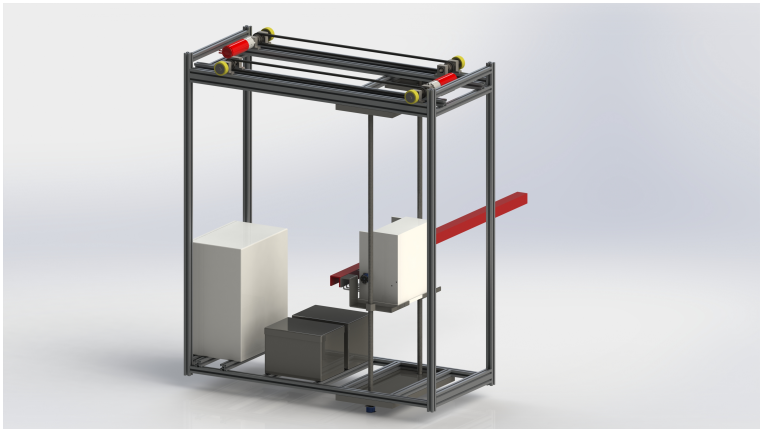


Figur 5: Oppbygging av bur oppsett

Det skal være en fôringsarm som holder seg over alle 10 burene på rekken. Det som skal skje er at den skal fore alle 10 burene.

2.3 Styring

Det er en vogn på siden som skal ha alt av styring for foring og oppfølgning av hummer. Foring av hummer vil skjedd igjennom en fôringsarm. Denne fôringsarmen har 10 stk dyser som gir ut maten. I tillegg på denne fôringsarmen er at det er 5 kamera. Tanken med dette er at de tar bilder på alle odde talls burene når den går opp over. Når du ta skal gå tilbake dyttes armen over fra odde talls rekkene til par talls rekkene, sånn at det blir tatt bilder på disse på tilbakeveien. Dette skjer fra en vogn med en foringsarm som vist i Figur 6



Figur 6: Styrevogn med arm

3 Komponenter

3.1 Programmerbar Logisk Styring [PLS]

Omron CJ1M-CPU11-ETN er den programmerbar logisk styring [PLS] komponenten som blir brukt i denne oppgaven. Dette er en PLS fra Omron som gjør at vi kan ha internet kontakt mellom PLS og PC, sånn at vi kan sende info fram og tilbake over internet kontakten.

Figur 7: Bilde av CJ1M-CPU11-ETN [2]

3.2 RaspberryPi 4

RaspberryPi 4 er en liten datamaskin. Den er nesten like liten som et kredit kort. I pakken som dette prosjekte har kjøp er med et kasse for å sette inn Raspberry PI inn i med en ekstra vifte. Noen av fordelene med en Raspberry pi 4 model B vs en Raspberry pi 3 model B er at generasjon 4 gir det en plass mellom 2 til 3 ganger så rask behandlingstid. I tillegg er genrasjon 4 satt opp med en bedre multi media muligheter. Dette er grunnet at kan kjøre 4k 60 frames per sekund.

Figur 8: Bilde av en Raspberry Pi 4 [8]

Denne har inngangs signaler fra eksterne kilder gjennom et 5v signal eller kan gi utgangs signaler på 5v eller gjennom releutganger. Ved bruk av releutganger kan man få opp til 230v på utgangen når relegår til ønsket posisjon. Det er både NO og NC utganger.

3.3 Python

Python er et objektbasert programmeringsspråk som gir mulighet for å lage program som kan bildebehandling. Det har muligheter for å behandle bilder på en sånn måtte at programmet kan den gjenkjenne ønsket bevegelse. Hvis det ikke blir gjort kan man da sende et tegn om at det ikke skal gå videre i koden.

3.4 USB Kamera

Problemstilling med oppgaven var å finne et USB kamera som kunne brukes til å ta bilder med. Dette kamera trenger å kunne ta bilde med av hummeret på nærthold. Dette er ikke mye plass mellom hyllene sånn at det må være plass besparende. Derfor trenger vi et kamera med kort fokus område.

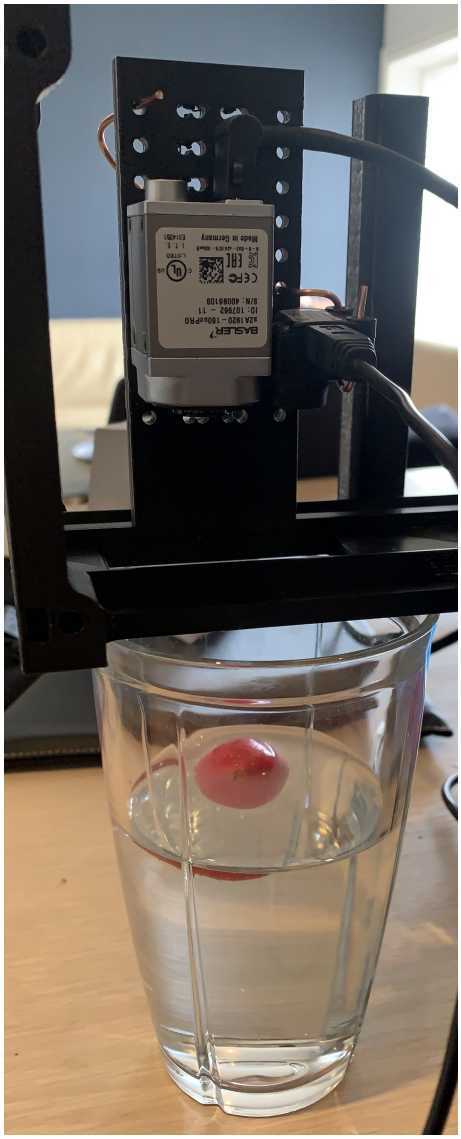
Derfor ble det sjekket hvilken muligheter det var på UiS med utstyr. UiS har noen IDS kamera av type UI-1007XS-C. Kamera er av kun farger sånn at det går ikke å sette den til svart/hvit

Figur 9: Bilde av USB camera fra IDS

Et kamera som Norwegian Lobster Farm har motatt i samarbeid grunnet EU støtten var Basler kamera. dette er av type a2A1920-160ucPRO. Kamera er av kun farger sånn at det går ikke an å sette den til svart/hvit

Figur 10: Bilde av USB camera fra Basler

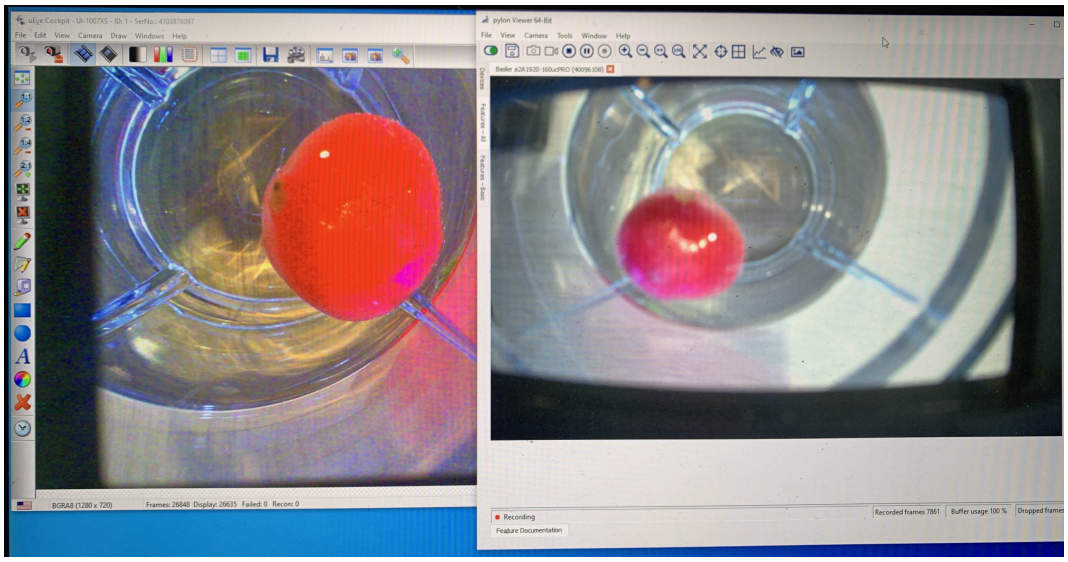
Bilde under viser hvordan den fysisk utforming av kamerane er mot hverandre. Dette gir en høyde forskjell som ble et problem ved installasjon av Basler kamera inn i riggen.



Figur 11: Visser hvordan sammeligningen av to kamera ble gjort

De neste to bildene viser sammenligning av fysisk utforming og hvordan bilde sammenligning.

Figur 12: Visser tydlig hvordan kamerane ble plassert ved sammeligningen av typene



Figur 13: Visser hvordan bildene blir fra de forskjellige kamerane

Tabellen under viser info som er knyttet til spesifikasjoner:

Kamera	Objekt fokus	Bildefrekvens	Oppløsning	Pris
IDS	Manuelt objekt justering	15	5.04 MP	199.00 EUR
Basler	Væske objekt objektiv	160	2.3 MP	419.00 EUR

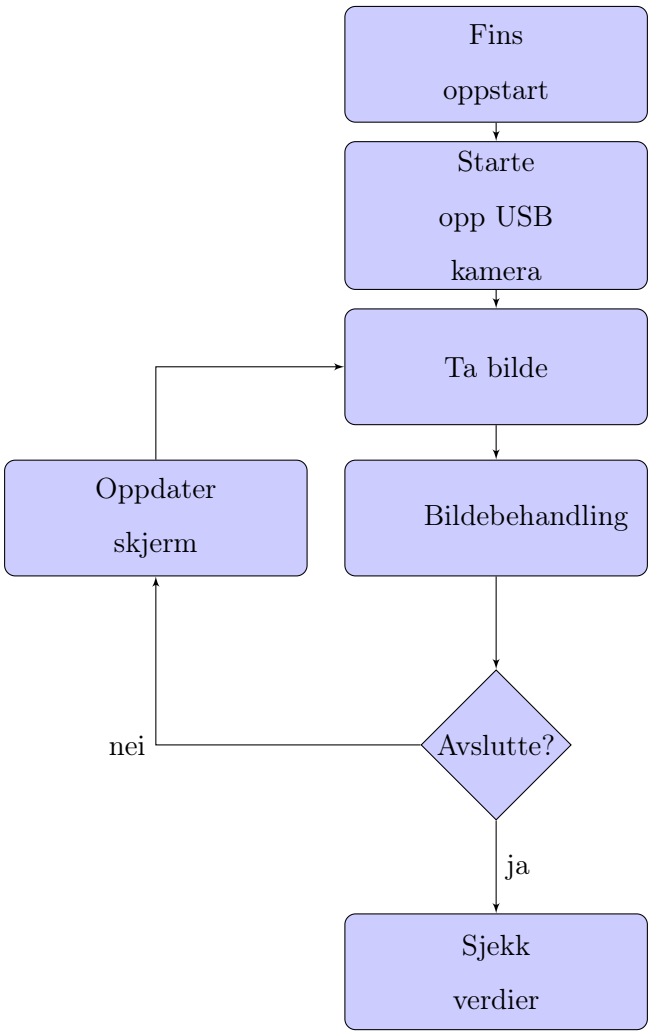
Tabell 1: Sammenligning med IDS kamera og Basler Kamera

Ut fra disse opplysningene gitt i dette kapitelet ble det tatt en valg om å bruke IDS på grunn av vi ikke trenger mer enn en bildefrekvens på 10 per kamera. Tillegg har prisen på kameraet på 199 EUR vært en utgjørende faktor når prosjektet trenger 5 stykker. Dette gir en prisforskjell på 1100 EUR mellom disse 2 produsentene. Den fysiske utformingen gjør også at det ble et plass problem på riggen hos Norweigan Lobster Farm

4 Programmering

Tankeflyten i framgangsmåten for problemstillingen er å først starte opp kommunikasjonen med PLS koden visses i kapitel 4.1. Etter at kommunikasjonen er sattopp blir USB kamera startet og klargjort gjennom koden som blir vist i kapittel 4.2. Når kameraene er sattopp sånn at de kan begynne å ta bilder, kan bildebehandlingen, som blir gjort gjennom denne koden kommer fram i 4.3. Ved bildebehandling kommer det en grade verdi på hummeren. Denne verdien på hummeren blir lagret i en fil som sjekker mot gårsdagens verdi, koden for dette blir hengitt i kapitel 4.4.

Under er et forenklet flytskjema av prosessen:



4.1 Omron FINS Driver

Koden i sin helhet kan finnes i vedlegg 7.1. Denne koden startet med utgangspunkt i koden fra aphyt [1].Under er koden for kommunikasjon mellom Python og PLS framgitt:
Koden starter med å lagge en forbindelse med PLS. For å kunne gjøre dette må man ha IP adressen til PLS og Destinasjons node nummeret til PLS enheten du skal bruke.

```
1 fins_instance = fins.udp.UDPFinsConnection()
2 fins_instance.connect('192.168.250.101')
3 fins_instance.dest_node_add=101
4 fins_instance.srce_node_add=0
```

Listing 1: Omron FINS Driver oppstart

Koden under viser hvordan å hente ut info. Dette er for å gi info inn i koden som gjør at kan brukes til forskjellige ting. Eksempelvis for å gi info som kan brukes i den grafiske framvisningen på skjermen knyttet til Raspberry PI. Det kan hentes ut enten et bit eller et helt ord. Linje 1 under viser et helt ord mens linje 2 henter kun et bit.

```
1 fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().DATA_MEMORY_WORD,b
    '/x00/x40/x00')
2 fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().WORK_BIT,b'/x00/
    x00/x02'):
```

Listing 2: Omron FINS Driver bit og word

4.2 USB kamera kode

Koden i sin helhet kan finnes i vedlegg 7.1. Under er koden for hvordan å starte opp USB kamera fra IDS framgitt:
Her finner vi ut hvor mange kamera det er koblet til Raspberry PI. Dette er for å vite hvor mange kamera vi skal initialisere.

```
maxCamera = ueye.INT()
nRet = ueye.is_GetNumberOfCameras(maxCamera)
```

Listing 3: Finne antall kamera

Koden under setter hvilket kamera vi skal initialisere og setter bilde størrelsen til 640x480. For løkken brukes for å initialisere kamerane i en økende rekkefølge etter IDen til kamerane og bilde størrelsen er satt for å minske minne bruken på Raspberry.

```
hCam = ueye.HIDS(i+1)
rectAOI = ueye.IS_RECT()
rectAOI.s32X = ueye.INT(0)
rectAOI.s32Y = ueye.INT(0)
rectAOI.s32Width = ueye.INT(640)
rectAOI.s32Height = ueye.INT(480)
```

Listing 4: Sette verdier

Koden under er hvordan vi snakker med kamera. uEye biblioteket har instrukser med is som prefiks. De returnerer en Integer som forteler om operasjonen ble gjennomført eller ikke. I eksemplet tar vi å initialiserer kamera og binder det til programmet og hvis operasjonen ikke ble gjennomført skriver vi ut en feilmelding.

```
nRet = ueye.is_InitCamera(hCam, None)
if nRet != ueye.IS_SUCCESS:
    print("is_InitCamera ERROR")
```

Listing 5: Setter opp kamera

I koden under hentes bilde fra minnet til Raspberry PI, gjør det om til en NumPy matrise som CV2 klarer å jobbe med og setter hvilket kamera det er oppi venstre hjørnet av bilde. Dette er for å kunne vise bilde på skjermen og at det skal være tydelig hvilket kamera man ser på.

```
array = ueye.get_data(camera['pcMemory'], camera['width'], camera['height'
    ], nBitsPerPixel, pitch, copy=False)
frame = np.reshape(array, (height.value, width.value, camera['
    bytes_per_pixel']))
cv2.putText(frame,str(camera['img_counter']),
        topLeftCornerOfText,
        font,
        fontScale,
        fontColor,
        lineType)
```

Listing 6: Lager en NumPy matrise som CV2 klarer å vise på skjermen

Første linje i koden under setter skalering på bildet som skal vises i vinduet så det ikke blir større en skjermen. Andre linje tar to matriser og hvilken akse den skal sette sammen for å bygger den sammen til en matrise. Linje tre lager et svart bilde som vi bruker til å fylle inn i matrisen siden det trengs like stor y-akse i matrisen for å foreksempel lage en 2x3 bilde med bare 5 kamera.

```
frame = cv2.resize(frame, (0, 0), fx=scale, fy=scale)
view = np.concatenate((view, frame), axis=1)
img = np.zeros((height.value,width.value,3), np.uint8)
```

Listing 7: Sette alle bildene sammen til ett stort bilde

Neste del av koden tar matrisen og en String som variabler og lager eller oppdaterer et vindu på skjermen. Dette er for å vise frem bilde til operatør.

```
cv2.imshow(hei, view)
```

Listing 8: Vise bilde på skjermen

Koden under lagrer hvert enkelt bilde. Den starter ved å sjekke at mappen den skal lagre i eksisterer med funksjonen makeDir, hvis den ikke eksisterer, lages mappen. Koden tar så å lagrer bildet i mappe strukturen før den sender bilde til bilde behandling som forklares i kapittel 4.3

```
bsv.makeDir(hylle, rad, camera['img_counter'])

img_name = "{}{}{}{}.jpg".format(hylle,rad,camera['img_counter'], now.
    strftime("%d.%m.%Y_%H.%M:%S"))
cv2.imwrite(img_name, tmp[i])
print("{} written!".format(img_name))
bsv.calculateAngel(hylle, rad, camera['img_counter'], img_name)
```

Listing 9: Lagre bilde

Siste kodesegmentet frigjør først minne på Raspberry PI som har blitt brukt før den kutter linken mellom programmet og kamera. Etter at alle kameraene har blitt frigjort, så destruerer vi alle vinduer som vi har laget med CV2, så det ikke henger noe igjen etter at programmet er ferdig.

```
ueye.is_FreeImageMem(camera['hCam'], camera['pcMemory'], camera['MemID'])
ueye.is_ExitCamera(camera['hCam'])
cv2.destroyAllWindows()
```

Listing 10: Frigjøre minne og kamera etter kjøring

4.3 Bilde behandling

Koden i sin helhet for denne delen finnes i vedlegg 7.3. Under er koden segmenter som viser viktig ting som skjer for å vise bildebehandlingen. Bilder under er av en hummer gitt av Norwegian Lobster Farm. Denne var ikke i produksjonen under oppgaven grunnen kun små hummer under start av oppgaven.

Figur 14: Original bilde av hummeren

Koden under er en funksjon som regner ut radianen på et sett med koordinater. Der X1 og Y1 er det ene punktet mens X2 og Y2 er det andre punktet.

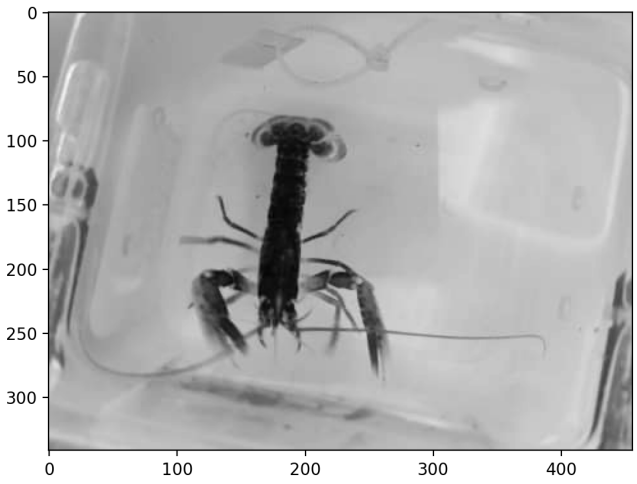
```
1 def myradians(X1, Y1, X2, Y2):
2     myradians = math.atan2(Y1-Y2, X1-X2)
3     return myradians
```

Listing 11: Utregning av radianen til to punkter

Under er det en kode segment på å gjøre om bildet fra farger til svart hvit.

```
1 def calculateAngel(hylle, rad, bur, imageName):
2     img = cv2.imread(imageName,0)
3     ret, mask = cv2.threshold(img, 60, 120, cv2.THRESH_BINARY_INV)
```

Listing 12: Omgjøring av farget bilde til svart hvit

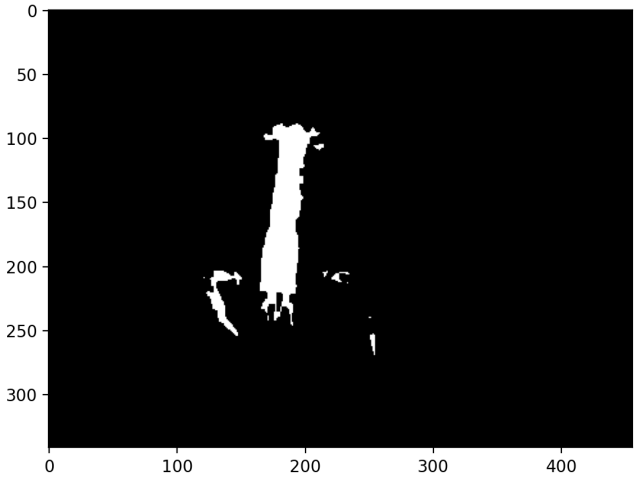


Figur 15: Bilde av svart hvit av hummeren

Under er kodesegment på å gjøre om bildet fra svart hvit til piksel lært bilde svart eller hvit. Her blir det ingen grå toner brukt siden bilde skal være av 1 eller 0.

```
1    kernel = np.ones((5,3),np.uint8)
2    mask = cv2.erode(mask,kernel,iterations = 1)
```

Listing 13: Omgjøring svart hvit bilde til piksel basert bilde.



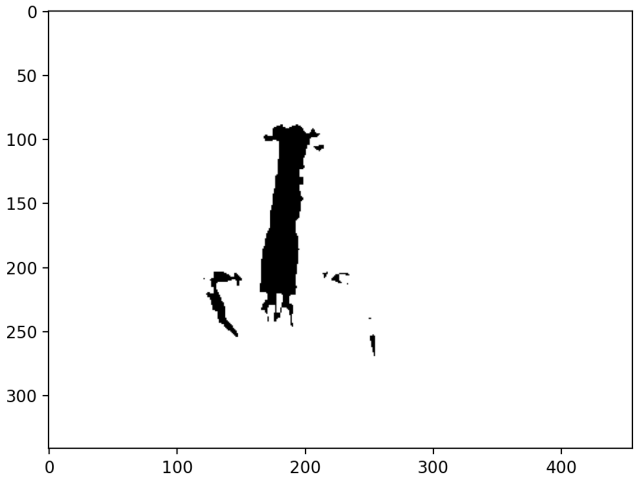
Figur 16: Bilde av svart eller hvit piksler av hummer

Koden under er å bytte mellom 1 og 0 fra forrige segment. Dette gjør at hummeren blir

svart i bilde sånn at det skal bli letter å finne retningen på dette.

```
    mask = cv2.erode(mask,kernel,iterations = 1)
    mat = np.argwhere(mask != 0)
    mat[:, [0, 1]] = mat[:, [1, 0]]
    mat = np.array(mat).astype(np.float32) #have to convert type for PCA
```

Listing 14: Bytter om på svart hvit



Figur 17: Bilde innovert bilde av figur 16 dette gi en svart hummer som brukes som objekt.

Under er kode for å finne retningen på hummeren. Dette er for å finne ut hvilken grad det er på hummeren i forhold til hans plassering i buret.

```
    try:
        m, e = cv2.PCACompute(mat, mean=np.array([]))
    except:
        print("could not calculate angel")
        return
    center = tuple(m[0])
    endpoint = tuple(m[0] + e[0]*100)

    cv2.circle(mask, center, 5, 255)
    cv2.line(mask, center, endpoint, 255)
```

Listing 15: Lager koordinater på objektet.

Hvordan utregningen på radianer, som var i første segmentet, i dette kapitelet 4.3, er vist i koden under. Koden gjør det om til grader. Dette er fordi at vi skal opp gi vinkelen på hummeren.

```
1    radin = myradians(center[0],center[1],endpoint[0],endpoint[1])
2    degree = math.degrees(radin)
```

Listing 16: Regne ut først radianer og der etter det om til grader

Under er det segmentet som gjør om verdiene fra funksjonen om til String verdier, sånn at de kan senere brukes til å gi navn i JSON filen.

```
1    h = str(hylle)
2    r = str(rad)
3    b = str(bur)
```

Listing 17: Omgjøring av verdier til string

Koden under tar først å sjekker om 'angels' filen eksisterer. Hvis den ikke finner filen, lager den et objekt med strukturen vist under. Dette objektet lagres etterpå som en JSON dokument.

h	Hylle nummer
r	Rad nummer
b	Bur nummer
new	Ny vinkel
old	vinkel fra forrige utregning av vinkel

Tabell 2: Data strukturen i JSON filen

Hvis 'angels' filen eksisterer laster den inn filen som et objekt. Koden sjekker så om denne kombinasjonen av hylle, rad og bur har blitt lagret før. Dersom det det er første gangen dette burdet har fått en vinkel utregnet lagres den i 'new', og har buret allerede en gammel vinkel i 'new' tar den å flytter den over i old. Dette er for å kunne sammenligne senere og si ifra hvis hummeren har flyttet på seg eller ikke. Koden avslutter med å overskrive det gamle dokumentet med den nye informasjonen.

```
    if not os.path.exists("cage_angels.json"):
        data = {}
        data[h] = {}
        data[h][r] = {}
        data[h][r][b] = {'new':degree}
        with open('cage_angels.json', 'w') as outfile:
            json.dump(data, outfile)
    else:
        with open('cage_angels.json', 'r+') as json_file:
            data = json.load(json_file)
            if hylle not in data:
                data[h] = {}
            if rad not in data[h]:
                data[h][r] = {}
            if bur not in data[h][r]:
                data[h][r][b] = {'new':degree}
            else:
                data[h][r][b]['old'] = data[h][r][b]['new']
                data[h][r][b]['new'] = degree
            json_file.seek(0)
            json_file.truncate(0)
            json.dump(data, json_file)
```

Listing 18: CalculateAngel

4.4 Sammenligning av grader mellom i går og i dag

Koden i sin helhet kan finnes i vedlegg 7.3.

Koden starter først med å sjekke at JSON filen som har alle viklene eksisterer. Hvis filen ikke eksiterer kommer en popup med feilmelding. Koden tar så å laster inn filen som et objekt ved hjelp fra JSON biblioteket før den jobber gjennom hele objektet for å se om hvert bur sin 'old' og 'new' vinkel verdig er lik. Hvis de er like tar koden å legger identifikatoren til hvilket bur det er og lagrer det i 'warn' variabelet.

```
if not os.path.exists("cage_angels.json"):
    alert_popup("Error", "cage_angles.json does not exist")
    return
with open('cage_angels.json') as file:
    data = json.load(file)
    for hylle in data:
        for rad in data[hylle]:
            for bur in data[hylle][rad].keys():
                if data[hylle][rad][bur]['old'] == data[hylle][rad][bur]['
    new']:
                    warn.append(hylle + ":" + rad + ":" + bur)
```

Listing 19: Sjekke dagens vinkel mot forrige

Koden under ser om det er lagert data i 'warn' og skriver det ut hvis det er noe der.

```
if warn:
    message = ""
    for bur in warn:
        message += bur + ", "

    alert_popup("Se  p   bur", message)
```

Listing 20: Beskjed til operatør

5 Diskusjon/konkusjon

Arbeidet som er dokumentert i rapporten, som er basert på å lage kommunikasjon og bildebehandling mellom en Raspberry PI og en PLS. Dette for å lagre bilder i en mappe sånn at disse kan bli brukt til å gi en historikk på hummerene.
Framgangsmåten som ble brukt for å løse oppgaven er som følger:
1. Sette meg inn i komponenter som skulle brukes.
2. Finne ut hva som var den best måte å kommunisere med PLS på.
3. Finne en enhet som kunne programmere Python på og som enkel kunne plasseres i et styrings skap.
4. Valg av USB kamera.
5. Lage kode for å utføre jobben med bildebehandling og kommunikasjon.
6. Utføre test på Norwegian Lobster Farm

I planleggingsarbeidet ble det fort klart at det er flere måter å løse dette på. Alternativene for komponenter er mange, og det er flere leverandører som kan gi kamera eller små PC/-mikrokontrollere. Det har blitt prøvd å løse denne problemstillingen på en sånn måtte at det blir enkel, oversiktlig og kostnadseffektivt.
Omron PLS ble valgt med bakrun i hvilket utstyr som allerede eksisterete i styringskapet til Norwegian Lobster Farm. Derfor var det kun mulig å bruke FINS protokollen til å snakke mellom Omron PLS og PC enheten.
RaspberryPi er en billig utviklings PC med mye funksjonalitet og programmeringsmuligheter. Her har du både WIFI tilkobling for å kunne flytte ut bilder og en Ethernet tilkobling, samt USB 3 som gjør at vi kan bruke en USB hub for å få alle 5 kamerane inn på Raspberry PI.
IDS sine kamera ble valgt på grunn av fysisk utforming, pris og oppfylte kravet om bildefrekvens.
Denne problemstillingen er av en sånn art at du kan videreutvikle den for å tilpasse seg til nye behov hos Norwegian Lobster Farm.

5.1 Forbedringer

I denne framgangsmåten er det brukt en Omron CJ1M-PLS istedenfor noen av de nyere modelene til Omron. Med nyere modeler hadde det vært mulig for enklere kommunikasjon enn å gå via FINS protokollen. Da ville vi enklere kunne flytte foringsarmen til en spesifikk rad.

Videre utvikle bildebehandlingen for å funke for små hummer som akkurat har flyttet inn i egne bur.

Bygge en mini PC som kan gi flere USB busser for mer kommunikasjon mellom kamera og mini PC. Per i dag kan ikke Raspberry PI klare mer enn 5 kamera.

En av problemstilling som har kommet fram under oppgaven er at det ikke er et godt støtte bibliotek til Python for IDS kamera. Dette er et C++ bibliotek som er gjort om til Python. Programmet må designs rundt dette, og det kan det være lurt å vurdere å gå over til C++ istedenfor Python.

6 Referanser

Referanser

[1] aphyt. Omron fins driver in python 3. https://www.aphyt.com/index.php/omron-fins-driver-in-python-3.

[2] PLC DEV. Cj1m-cpu11-etn. http://www.plcdev.com/parts/omron-sysmaccj1-cj1m-cpu11-etn.html.

[3] ERA-LEARN. Project: Demonstration and validation of a novel approach to european lobster aquaculture. https://www.era-learn.eu/network-information/networks/eurostars-2/eurostars-cut-off-5/demonstration-and-validation-of-a-novel-approach-to-european-lobster-aquaculture.

[4] Norwegian Lobster Farm. Automarus – grant agreement no 880911. https://www.norwegian-lobster-farm.com/automarus-grant-agreement/.

[5] Norwegian Lobster Farm. Norwegian lobster farm får eurostarprosjekt. https://www.norwegian-lobster-farm.com/norwegian-lobster-farm-far-eurostarprosjekt/.

[6] Norwegian Lobster Farm. Produksjonsmetode. https://www.norwegian-lobster-farm.com/kategori/produksjonsmetode/.

[7] Norwegian Lobster Farm. Underliggende teknologi. https://www.norwegian-lobster-farm.com/teknologi/.

[8] TechRepublic. The raspberry pi 4 model b. https://www.zdnet.com/article/what-is-the-raspberry-pi-4-everything-you-need-to-know-about-the-tiny-low-cost-computer.

7 Vedlegg

7.1 Main.py

```
# ===========================================================================#
#                                                                            #
#  Copyright (C) 2006 - 2018                                                 #
#  IDS Imaging Development Systems GmbH                                      #
#  Dimbacher Str. 6-8                                                        #
#  D-74182 Obersulm, Germany                                                 #
#                                                                            #
#  The information in this document is subject to change without notice      #
#  and should not be construed as a commitment by IDS Imaging Development    #
#  Systems GmbH. IDS Imaging Development Systems GmbH does not assume any    #
#  responsibility for any errors that may appear in this document.           #
#                                                                            #
#  This document, or source code, is provided solely as an example           #
#  of how to utilize IDS software libraries in a sample application.         #
#  IDS Imaging Development Systems GmbH does not assume any                  #
#  responsibility for the use or reliability of any portion of this          #
#  document or the described software.                                       #
#                                                                            #
#  General permission to copy or modify, but not for profit, is hereby       #
#  granted, provided that the above copyright notice is included and         #
#  reference made to the fact that reproduction privileges were granted      #
#  by IDS Imaging Development Systems GmbH.                                  #
#                                                                            #
#  IDS Imaging Development Systems GmbH cannot assume any responsibility     #
#  for the use or misuse of any portion of this software for other than      #
#  its intended diagnostic purpose in calibrating and testing IDS            #
#  manufactured cameras and software.                                        #
#                                                                            #
# ===========================================================================#
# Developer Note: I tried to let it as simple as possible.
# Therefore there are no functions asking for the newest driver software or
    freeing memory beforehand, etc.
# The sole purpose of this program is to show one of the simplest ways to
   interact with an IDS camera via the uEye API.
# (XS cameras are not supported)
```

```
34 # ------------------------------------------------------------------------
35
36 # Libraries
37 from pyueye import ueye
38 import numpy as np
39 import cv2
40 import os
41 import bilde_sjekk_og_vinkel as bsv
42 import fins.udp
43 import time
44 import popupWindow as pw
45 from datetime import datetime
46
47 fins_instance = fins.udp.UDPFinsConnection()
48 fins_instance.connect('192.168.250.101')
49 fins_instance.dest_node_add=101
50 fins_instance.srce_node_add=0
51
52 rectAOI = ueye.IS_RECT()
53 pitch = ueye.INT()
54 nBitsPerPixel = ueye.INT(32)
55
56 maxCamera = ueye.INT()
57 nRet = ueye.is_GetNumberOfCameras(maxCamera)
58
59 cameras = []
60 # Kan skrive in hvor mange kamera, eller bruke maxCamera for    bruke alle
       som er tilkoblet.
61 cameraAmount = 5
62 rowOfCages = 120
63
64 # hvis du bruker maxCamera  m   du legge til .value  p   cameraAmount
65 for i in range(cameraAmount):
66     hCam = ueye.HIDS(i+1)
67     sInfo = ueye.SENSORINFO()
68     cInfo = ueye.CAMINFO()
69     MemID = ueye.int()
70     pcImageMemory = ueye.c_mem_p()
71     rectAOI = ueye.IS_RECT()
72     rectAOI.s32X = ueye.INT(0)
73     rectAOI.s32Y = ueye.INT(0)
```

```
74      rectAOI.s32Width = ueye.INT(640)
75      rectAOI.s32Height = ueye.INT(480)
76      pitch = ueye.INT()
77      nBitsPerPixel = ueye.INT(24)  # 24: bits per pixel for color mode; take
         8 bits per pixel for monochrome
78      channels = 3  # 3: channels for color mode(RGB); take 1 channel for
       monochrome
79      m_nColorMode = ueye.INT()  # Y8/RGB16/RGB24/REG32
80      bytes_per_pixel = int(nBitsPerPixel / 8)
81      nRet = ueye.is_InitCamera(hCam, None)
82      if nRet != ueye.IS_SUCCESS:
83          print("is_InitCamera ERROR")
84      nRet = ueye.is_GetCameraInfo(hCam, cInfo)
85      if nRet != ueye.IS_SUCCESS:
86          print("is_GetCameraInfo ERROR")
87      nRet = ueye.is_GetSensorInfo(hCam, sInfo)
88      if nRet != ueye.IS_SUCCESS:
89          print("is_GetSensorInfo ERROR")
90      nRet = ueye.is_ResetToDefault(hCam)
91      if nRet != ueye.IS_SUCCESS:
92         print("is_ResetToDefault ERROR")
93      nRet = ueye.is_SetColorMode(hCam, ueye.IS_CM_JPEG)
94      if nRet != ueye.IS_SUCCESS:
95         print("is_SetColorMode ERROR")
96      test = ueye.double()
97      nRet = ueye.is_SetFrameRate(hCam, ueye.double(1), test)
98      if nRet != ueye.IS_SUCCESS:
99         print("is_SetFrameRate ERROR")
100
101     nRet = ueye.is_SetDisplayMode(hCam, ueye.IS_SET_DM_DIB)
102     if int.from_bytes(sInfo.nColorMode.value, byteorder='big') == ueye.
       IS_COLORMODE_BAYER:
103         # setup the color depth to the current windows setting
104         ueye.is_GetColorDepth(hCam, nBitsPerPixel, m_nColorMode)
105         bytes_per_pixel = int(nBitsPerPixel / 8)
106
107     elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') == ueye.
       IS_COLORMODE_CBYCRY:
108         # for color camera models use RGB32 mode
109         m_nColorMode = ueye.IS_CM_BGR8_PACKED
110         nBitsPerPixel = ueye.INT(24)
```

```
111         bytes_per_pixel = int(nBitsPerPixel / 8)
112
113     elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') == ueye.
    IS_COLORMODE_MONOCHROME:
114         # for color camera models use RGB32 mode
115         m_nColorMode = ueye.IS_CM_MONO8
116         nBitsPerPixel = ueye.INT(8)
117         bytes_per_pixel = int(nBitsPerPixel / 8)
118
119     else:
120         # for monochrome camera models use Y8 mode
121         m_nColorMode = ueye.IS_CM_MONO8
122         nBitsPerPixel = ueye.INT(8)
123         bytes_per_pixel = int(nBitsPerPixel / 8)
124
125     #Skrur av autofokus
126     nRet = ueye.is_Focus(hCam, ueye.FOC_CMD_SET_DISABLE_AUTOFOCUS, ueye.INT
    (), ueye.INT(0))
127     if nRet != ueye.IS_SUCCESS:
128         print("is_Focus ERROR")
129
130     #Setter fokus p  kameraet
131     nRet = ueye.is_Focus(hCam, ueye.FOC_CMD_SET_MANUAL_FOCUS, ueye.INT(170)
    , ueye.INT(4))
132     if nRet != ueye.IS_SUCCESS:
133         print("is_Focus ERROR")
134
135     #Setter st relsen p  bilde vi skal ha
136     nRet = ueye.is_AOI(hCam, ueye.IS_AOI_IMAGE_SET_AOI, rectAOI, ueye.
    sizeof(rectAOI))
137     if nRet != ueye.IS_SUCCESS:
138         print("is_AOI ERROR")
139
140     width = rectAOI.s32Width
141     height = rectAOI.s32Height
142
143     nRet = ueye.is_AllocImageMem(hCam, width, height, nBitsPerPixel,
    pcImageMemory, MemID)
144     if nRet != ueye.IS_SUCCESS:
145         print("is_AllocImageMem ERROR")
146     else:
```

```
147         # Makes the specified image memory the active memory
148         nRet = ueye.is_SetImageMem(hCam, pcImageMemory, MemID)
149         if nRet != ueye.IS_SUCCESS:
150             print("is_SetImageMem ERROR")
151         else:
152             # Set the desired color mode
153             nRet = ueye.is_SetColorMode(hCam, m_nColorMode)
154     #Setter opp til    sende en str m av bilder til programet
155     nRet = ueye.is_CaptureVideo(hCam, ueye.IS_DONT_WAIT)
156     if nRet != ueye.IS_SUCCESS:
157         print("is_CaptureVideo ERROR")
158 
159     #Henter plasering av minne p   dataen til    se bildene p
160     nRet = ueye.is_InquireImageMem(hCam, pcImageMemory, MemID, width,
        height, nBitsPerPixel, pitch)
161     if nRet != ueye.IS_SUCCESS:
162         print("is_InquireImageMem ERROR")
163     else:
164         print("Camera",i+1,"Set up")
165     img_counter = (i*2)+1
166     d = dict()
167     d["pcMemory"] = pcImageMemory
168     d["width"] = width
169     d["height"] = height
170     d["hCam"] = hCam
171     d["MemID"] = MemID
172     d["rectAOI"] = rectAOI
173     d["pitch"] = pitch
174     d["nBitsPerPixel"] = nBitsPerPixel
175     d["bytes_per_pixel"] = bytes_per_pixel
176     d["m_nColorMode"] = m_nColorMode
177     d["img_counter"] = img_counter
178     cameras.append(d)
179 
180 setFPS = True
181 wayup = True
182 
183 font                   = cv2.FONT_HERSHEY_SIMPLEX
184 topLeftCornerOfText = (10,50)
185 bottomLeftCornerOfText = (10,100)
186 fontScale              = 1
```

```
fontColor              = (255,255,255)
lineType               = 2
scale = 0.9

while True:
    frameArray = 0
    tmp = []
    view = []
    view2 = []
    rad = fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    DATA_MEMORY_WORD,b'/x00/x40/x00')
    bildenummer = 0
    for camera in cameras:
        # In order to display the image in an OpenCV window we need to...
        # ...extract the data of our image memory
        array = ueye.get_data(camera['pcMemory'], camera['width'], camera['
    height'], nBitsPerPixel, pitch, copy=False)

        height = camera['height']
        width = camera['width']

        # ...reshape it in an numpy array...
        frame = np.reshape(array, (height.value, width.value, camera['
    bytes_per_pixel']))
        if wayup:
            cv2.putText(frame,str(camera['img_counter']),
                topLeftCornerOfText,
                font,
                fontScale,
                fontColor,
                lineType)
        else:
            cv2.putText(frame,str(camera['img_counter']+1),
                topLeftCornerOfText,
                font,
                fontScale,
                fontColor,
                lineType)

        # ...resize the image by a half
        frame = cv2.resize(frame, (0, 0), fx=scale, fy=scale)
```

```
225
226         #Lager et view array som skal vises p   skjermen
227         if tmp:
228             if len(tmp) < 3:
229                 view = np.concatenate((view, frame), axis=1)
230             elif len(tmp) == 3:
231                 view2 = frame
232             elif len(tmp) == 4:
233                 view2 = np.concatenate((view2, frame), axis=1)
234                 img = np.zeros((height.value,width.value,3), np.uint8)
235                 cv2.putText(img,'Bur nummer: ' + str(rad),
236                 topLeftCornerOfText,
237                 font,
238                 fontScale,
239                 fontColor,
240                 lineType)
241                 cv2.putText(img,'Bilde tatt av Bur nummer: ' + str(
    bildenummer),
242                 bottomLeftCornerOfText,
243                 font,
244                 fontScale,
245                 fontColor,
246                 lineType)
247                 img = cv2.resize(img, (0, 0), fx=scale, fy=scale)
248                 view2 = np.concatenate((view2, img), axis=1)
249                 view = np.concatenate((view, view2), axis=0)
250         else:
251             view = frame
252
253
254         #Setter hvor mange bilder i sekundet kameraet skal ta
255         if setFPS:
256             test = ueye.double()
257             nRet = ueye.is_SetFrameRate(camera['hCam'], ueye.double(10),
    test)
258             if nRet != ueye.IS_SUCCESS:
259                print("is_SetFrameRate ERROR")
260         tmp.append(frame)
261
262
263     #Lager og oppdaterer vinduet
```

```
264     hei = "SimpleLive_Python_uEye_OpenCV"
265         # ...and finally display it
266     cv2.imshow(hei, view)
267 
268     k = cv2.waitKey(1)
269     # Press esc if you want to end the loop
270     if k % 256 == 27 :
271         break
272     elif k % 256 == 13:
273         wayup = not wayup
274     elif k % 256 == 32 : # or  fins_instance.memory_area_read(fins.
    FinsPLCMemoryAreas().WORK_BIT,b'/x00/x00/x02'):
275         i = 0
276         hylle = 0
277          if fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    WORK_BIT,b'/x00/x01/x03'):
278             hylle = 1
279          elif fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    WORK_BIT,b'/x00/x01/x04'):
280             hylle = 2
281          elif fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    WORK_BIT,b'/x00/x01/x05'):
282             hylle = 3
283          else:
284             hylle = 10
285 
286         bit= fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    WORK_BIT,b'/x00/x01/x01')
287         bit2= fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    WORK_BIT,b'/x00/x01/x02')
288         now = datetime.now()
289         if bit == 1:
290             wayup = True
291         elif bit2 == 1:
292             wayup = False
293         if wayup: #bit for hvilken vei den kj rer
294             for camera in cameras:
295                 bsv.makeDir(hylle, rad, camera['img_counter'])
296 
297                 #Dette tar bilde fra hvert enkelt kamera.
```

```
298                 img_name = "{}{}{}{}.jpg".format(hylle,rad,camera['
    img_counter'], now.strftime("%d.%m.%Y_%H.%M:%S"))
299                 cv2.imwrite(img_name, tmp[i])
300                 print("{} written!".format(img_name))
301                 bsv.calculateAngel(hylle, rad, camera['img_counter'],
    img_name)
302                 #text file with a table of yesterdays input and add in
    todays.
303 
304                 i += 1
305             bildenummer = rad
306         else:
307             for camera in cameras:
308                 #Pr ver    lage ny mappe hvis bur mappen ikke eksisterer
309                 bsv.makeDir(hylle, rad, camera['img_counter']+1)
310 
311                 #Dette tar bilde fra hvert enkelt kamera.
312                 img_name = "opencv_frame_{}.jpg".format(camera['img_counter
    ']+1)
313                 cv2.imwrite(img_name, tmp[i])
314                 print("{} written!".format(img_name))
315                 bsv.calculateAngel(img_name)
316                 #text file with a table of yesterdays input and add in
    todays.
317 
318                 i += 1
319             bildenummer = rad
320     setFPS = False
321 
322 # ---------------------------------------------------------------------
323 
324 for camera in cameras:
325     # Releases an image memory that was allocated using is_AllocImageMem()
    and removes it from the driver management
326     ueye.is_FreeImageMem(camera['hCam'], camera['pcMemory'], camera['MemID'
    ])
327 
328     # Disables the hCam camera handle and releases the data structures and
    memory areas taken up by the uEye camera
329     ueye.is_ExitCamera(camera['hCam'])
330 
```

```
331 # Destroys the OpenCv windows
332 cv2.destroyAllWindows()
333
334 bsv.compareImages()
335
336 print()
337 print("END")
```

Listing 21: Main.py

7.2 Main-test.py

```
# ============================================================================#
#                                                                             #
#  Copyright (C) 2006 - 2018                                                  #
#  IDS Imaging Development Systems GmbH                                       #
#  Dimbacher Str. 6-8                                                         #
#  D-74182 Obersulm, Germany                                                  #
#                                                                             #
#  The information in this document is subject to change without notice       #
#  and should not be construed as a commitment by IDS Imaging Development     #
#  Systems GmbH. IDS Imaging Development Systems GmbH does not assume any     #
#  responsibility for any errors that may appear in this document.            #
#                                                                             #
#  This document, or source code, is provided solely as an example            #
#  of how to utilize IDS software libraries in a sample application.          #
#  IDS Imaging Development Systems GmbH does not assume any                   #
#  responsibility for the use or reliability of any portion of this           #
#  document or the described software.                                        #
#                                                                             #
#  General permission to copy or modify, but not for profit, is hereby        #
#  granted, provided that the above copyright notice is included and          #
#  reference made to the fact that reproduction privileges were granted       #
#  by IDS Imaging Development Systems GmbH.                                   #
#                                                                             #
#  IDS Imaging Development Systems GmbH cannot assume any responsibility      #
#  for the use or misuse of any portion of this software for other than       #
#  its intended diagnostic purpose in calibrating and testing IDS             #
#  manufactured cameras and software.                                         #
#                                                                             #
# ============================================================================#
# Developer Note: I tried to let it as simple as possible.
# Therefore there are no functions asking for the newest driver software or
    freeing memory beforehand, etc.
# The sole purpose of this program is to show one of the simplest ways to
   interact with an IDS camera via the uEye API.
# (XS cameras are not supported)
# ----------------------------------------------------------------------------

# Libraries
```

```
from pyueye import ueye
import numpy as np
import cv2
import fins.udp

#fins_instance = fins.udp.UDPFinsConnection()
#fins_instance.connect('192.168.250.101')
#fins_instance.dest_node_add=101
#fins_instance.srce_node_add=0


rectAOI = ueye.IS_RECT()
pitch = ueye.INT()
nBitsPerPixel = ueye.INT(32)

maxCamera = ueye.INT()
nRet = ueye.is_GetNumberOfCameras(maxCamera)

cameras = []
# Kan skrive in hvor mange kamera, eller bruke maxCamera for    bruke alle
    som er tilkoblet.
cameraAmount = 5
rowOfCages = 120

# hvis du bruker maxCamera  m   du legge til .value  p   cameraAmount
for i in range(cameraAmount):
    hCam = ueye.HIDS(i+1)
    sInfo = ueye.SENSORINFO()
    cInfo = ueye.CAMINFO()
    MemID = ueye.int()
    pcImageMemory = ueye.c_mem_p()
    rectAOI = ueye.IS_RECT()
    rectAOI.s32X = ueye.INT(0)
    rectAOI.s32Y = ueye.INT(0)
    rectAOI.s32Width = ueye.INT(640)
    rectAOI.s32Height = ueye.INT(480)
    pitch = ueye.INT()
    nBitsPerPixel = ueye.INT(24)  # 24: bits per pixel for color mode; take
     8 bits per pixel for monochrome
    channels = 3  # 3: channels for color mode(RGB); take 1 channel for
    monochrome
```

```
    m_nColorMode = ueye.INT()  # Y8/RGB16/RGB24/REG32
    bytes_per_pixel = int(nBitsPerPixel / 8)
    nRet = ueye.is_InitCamera(hCam, None)
    if nRet != ueye.IS_SUCCESS:
        print("is_InitCamera ERROR")
    nRet = ueye.is_GetCameraInfo(hCam, cInfo)
    if nRet != ueye.IS_SUCCESS:
        print("is_GetCameraInfo ERROR")
    nRet = ueye.is_GetSensorInfo(hCam, sInfo)
    if nRet != ueye.IS_SUCCESS:
        print("is_GetSensorInfo ERROR")
    nRet = ueye.is_ResetToDefault(hCam)
    if nRet != ueye.IS_SUCCESS:
       print("is_ResetToDefault ERROR")
    nRet = ueye.is_SetColorMode(hCam, ueye.IS_CM_JPEG)
    if nRet != ueye.IS_SUCCESS:
       print("is_SetColorMode ERROR")
    test = ueye.double()
    nRet = ueye.is_SetFrameRate(hCam, ueye.double(1), test)
    if nRet != ueye.IS_SUCCESS:
       print("is_SetFrameRate ERROR")

    nRet = ueye.is_SetDisplayMode(hCam, ueye.IS_SET_DM_DIB)
    m_nColorMode = ueye.IS_CM_BGR8_PACKED
    nBitsPerPixel = ueye.INT(24)
    bytes_per_pixel = int(nBitsPerPixel / 8)

    #Skrur av autofokus
    nRet = ueye.is_Focus(hCam, ueye.FOC_CMD_SET_DISABLE_AUTOFOCUS, ueye.INT
    (), ueye.INT(0))
    if nRet != ueye.IS_SUCCESS:
        print("is_Focus ERROR")

    #Setter fokus p  kameraet
    nRet = ueye.is_Focus(hCam, ueye.FOC_CMD_SET_MANUAL_FOCUS, ueye.INT(170)
    , ueye.INT(4))
    if nRet != ueye.IS_SUCCESS:
        print("is_Focus ERROR")

    #Setter st relsen p  bilde vi skal ha
```

```
113     nRet = ueye.is_AOI(hCam, ueye.IS_AOI_IMAGE_SET_AOI, rectAOI, ueye.
        sizeof(rectAOI))
114     if nRet != ueye.IS_SUCCESS:
115         print("is_AOI ERROR")
116 
117     width = rectAOI.s32Width
118     height = rectAOI.s32Height
119 
120     nRet = ueye.is_AllocImageMem(hCam, width, height, nBitsPerPixel,
        pcImageMemory, MemID)
121     if nRet != ueye.IS_SUCCESS:
122         print("is_AllocImageMem ERROR")
123     else:
124         # Makes the specified image memory the active memory
125         nRet = ueye.is_SetImageMem(hCam, pcImageMemory, MemID)
126         if nRet != ueye.IS_SUCCESS:
127             print("is_SetImageMem ERROR")
128         else:
129             # Set the desired color mode
130             nRet = ueye.is_SetColorMode(hCam, m_nColorMode)
131     #Setter opp til    sende en str m av bilder til programet
132     nRet = ueye.is_CaptureVideo(hCam, ueye.IS_DONT_WAIT)
133     if nRet != ueye.IS_SUCCESS:
134         print("is_CaptureVideo ERROR")
135 
136     #Henter plasering av minne p   dataen til    se bildene p
137     nRet = ueye.is_InquireImageMem(hCam, pcImageMemory, MemID, width,
        height, nBitsPerPixel, pitch)
138     if nRet != ueye.IS_SUCCESS:
139         print("is_InquireImageMem ERROR")
140     else:
141         print("Camera",i+1,"Set up")
142     img_counter = (i*2)+1
143     d = dict()
144     d["pcMemory"] = pcImageMemory
145     d["width"] = width
146     d["height"] = height
147     d["hCam"] = hCam
148     d["MemID"] = MemID
149     d["rectAOI"] = rectAOI
150     d["pitch"] = pitch
```

```
151     d["nBitsPerPixel"] = nBitsPerPixel
152     d["bytes_per_pixel"] = bytes_per_pixel
153     d["m_nColorMode"] = m_nColorMode
154     d["img_counter"] = img_counter
155     cameras.append(d)
156 
157 setFPS = True
158 wayup = True
159 
160 font                   = cv2.FONT_HERSHEY_SIMPLEX
161 topLeftCornerOfText = (10,50)
162 bottomLeftCornerOfText = (10,100)
163 fontScale              = 1
164 fontColor              = (255,255,255)
165 lineType               = 2
166 scale = 0.9
167 
168 while True:
169     frameArray = 0
170     tmp = []
171     view = []
172     view2 = []
173     rad = 0 # fins_instance.memory_area_read(fins.FinsPLCMemoryAreas().
    DATA_MEMORY_WORD,b'/x00/x40/x00')
174     bildenummer = 0
175     for camera in cameras:
176         # In order to display the image in an OpenCV window we need to...
177         # ...extract the data of our image memory
178         array = ueye.get_data(camera['pcMemory'], camera['width'], camera['
    height'], nBitsPerPixel, pitch, copy=False)
179 
180         height = camera['height']
181         width = camera['width']
182 
183         # ...reshape it in an numpy array...
184         frame = np.reshape(array, (height.value, width.value, camera['
    bytes_per_pixel']))
185         if wayup:
186             cv2.putText(frame,str(camera['img_counter']),
187                 topLeftCornerOfText,
188                 font,
```

```
                fontScale,
                fontColor,
                lineType)
        else:
            cv2.putText(frame,str(camera['img_counter']+1),
                topLeftCornerOfText,
                font,
                fontScale,
                fontColor,
                lineType)

        # ...resize the image by a half
        frame = cv2.resize(frame, (0, 0), fx=scale, fy=scale)

        #Lager et view array som skal vises p  skjermen
        if tmp:
            if len(tmp) < 3:
                view = np.concatenate((view, frame), axis=1)
            elif len(tmp) == 3:
                view2 = frame
            elif len(tmp) == 4:
                view2 = np.concatenate((view2, frame), axis=1)
                img = np.zeros((height.value,width.value,3), np.uint8)
                cv2.putText(img,'Bur nummer: ' + str(rad),
                topLeftCornerOfText,
                font,
                fontScale,
                fontColor,
                lineType)
                img = cv2.resize(img, (0, 0), fx=scale, fy=scale)
                view2 = np.concatenate((view2, img), axis=1)
                view = np.concatenate((view, view2), axis=0)
        else:
            view = frame


        #Setter hvor mange bilder i sekundet kameraet skal ta
        if setFPS:
            test = ueye.double()
            nRet = ueye.is_SetFrameRate(camera['hCam'], ueye.double(10),
    test)
```

```
229             if nRet != ueye.IS_SUCCESS:
230                 print("is_SetFrameRate ERROR")
231         tmp.append(frame)
232 
233 
234     #Lager og oppdaterer vinduet
235     hei = "Kamera til hummer"
236         # ...and finally display it
237     cv2.imshow(hei, view)
238 
239     k = cv2.waitKey(1)
240     # Press esc if you want to end the loop
241     if k % 256 == 27 :
242         break
243     elif k % 256 == 13:
244         wayup = not wayup
245     setFPS = False
246 
247 # ---------------------------------------------------------------------------
248 
249 for camera in cameras:
250     # Releases an image memory that was allocated using is_AllocImageMem()
    and removes it from the driver management
251     ueye.is_FreeImageMem(camera['hCam'], camera['pcMemory'], camera['MemID'
    ])
252 
253     # Disables the hCam camera handle and releases the data structures and
    memory areas taken up by the uEye camera
254     ueye.is_ExitCamera(camera['hCam'])
255 
256 # Destroys the OpenCv windows
257 cv2.destroyAllWindows()
258 
259 print()
260 print("END")
```

Listing 22: Main-test.py

7.3 Bilde-sjekk-og-vinkel.py [BSV]

```
# Libraries
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import json
from tkinter import *

def alert_popup(title, message):
    """Generate a pop-up window for special messages."""
    root = Tk()
    root.title(title)
    w = 400     # popup window width
    h = 200     # popup window height
    sw = root.winfo_screenwidth()
    sh = root.winfo_screenheight()
    x = (sw - w)/2
    y = (sh - h)/2
    root.geometry('%dx%d+%d+%d' % (w, h, x, y))
    m = message
    w = Label(root, text=m, width=120, height=10)
    w.pack()
    b = Button(root, text="OK", command=root.destroy, width=10)
    b.pack()
    mainloop()

def show(img):
    plt.imshow(img, cmap="gray")
    plt.show()

def myradians(X1, Y1, X2, Y2):
    myradians = math.atan2(Y1-Y2, X1-X2)
    return myradians

def calculateAngel(hylle, rad, bur, imageName):
    img = cv2.imread(imageName,0)

```

```
    ret, mask = cv2.threshold(img, 60, 120, cv2.THRESH_BINARY_INV)  # turn
    60, 120 for the best OCR results
    kernel = np.ones((5,3),np.uint8)
    mask = cv2.erode(mask,kernel,iterations = 1)
    mat = np.argwhere(mask != 0)
    mat[:, [0, 1]] = mat[:, [1, 0]]
    mat = np.array(mat).astype(np.float32) #have to convert type for PCA

    try:
        m, e = cv2.PCACompute(mat, mean=np.array([]))
    except:
        print("could not calculate angel")
        return
    center = tuple(m[0])
    endpoint = tuple(m[0] + e[0]*100)

    cv2.circle(mask, center, 5, 255)
    cv2.line(mask, center, endpoint, 255)

    radin = myradians(center[0],center[1],endpoint[0],endpoint[1])
    degree = math.degrees(radin)

    h = str(hylle)
    r = str(rad)
    b = str(bur)

    if not os.path.exists("cage_angels.json"):
        data = {}
        data[h] = {}
        data[h][r] = {}
        data[h][r][b] = {'new':degree}
        with open('cage_angels.json', 'w') as outfile:
            json.dump(data, outfile)
    else:
        with open('cage_angels.json', 'r+') as json_file:
            data = json.load(json_file)
            if hylle not in data:
                data[h] = {}
            if rad not in data[h]:
                data[h][r] = {}
            if bur not in data[h][r]:
```

```
                data[h][r][b] = {'new':degree}
            else:
                data[h][r][b]['old'] = data[h][r][b]['new']
                data[h][r][b]['new'] = degree
            json_file.seek(0)
            json_file.truncate(0)
            json.dump(data, json_file)

def compareImages():
    warn = []
    if not os.path.exists("cage_angels.json"):
        alert_popup("Error", "cage_angles.json does not exist")
        return
    with open('cage_angels.json') as file:
        data = json.load(file)
        for hylle in data:
            for rad in data[hylle]:
                for bur in data[hylle][rad].keys():
                    if data[hylle][rad][bur]['old'] == data[hylle][rad][bur
    ]['new']:
                        warn.append(hylle + ":" + rad + ":" + bur)
    if warn:
        message = ""
        for bur in warn:
            message += bur + ", "

        alert_popup("Se  p   bur", message)

def makeDir(shelf, rad, dirName):
    mappe = shelf + rad + dirName + ""
    if not os.path.exists(mappe):
        os.mkdir(mappe)
```

Listing 23: Bilde-sjekk-og-vinkel.py