# Oculus FPV

## Prosjektrapport


Gruppe 5

Erik Hals
Jacob Prescott
Mats Krüger Svensson
Mads Falmår Wilthil



TPG4850 VR-Landsbyen
Våren 2014

NTNU

# Forord

Denne rapporten, sammen med en demonstrasjon og video, er vår besvarelse på prosjektdelen av faget *TPG4850 - Eksperter i Team VR-landsbyen*.

Problemet vi har jobbet med var *Oppgave 1: Bruk av UAVs i petroleumsbransjen* som vi valgte å løse ved og se på mulighetene for en *immersive* flyopplevelse med Virtual Reality-briller.

Vi ønsker å takke Åge Sivertsen for lån av elektroverksted, god hjelp til bygging av kretskort og tilgang på elektriske komponenter. Vi ønsker også å takke alle de som har hjulpet oss og gitt oss tilgang til det mekaniske verkstedet. Til slutt ønsker vi å takke Egil Tjåland for veiledning og motivasjon.

Erik Hals

Jacob Prescott

Mats Krüger Svensson

Mads Falmår Wilthil

# Innhold

# 1 Innledning

## 1.1 Dagens situasjon

UAVs, populært kalt droner, har den siste tiden gjort stort inntog som hobby, men også i næringslivet. De siste årenes utvikling på batterifronten har gjort at de får en brukbar flytid, og nye kontrollkort har gjort dem svært stabile og lette å fly. Mobilkamerarevolusjonen har ført til at små kamerabrikker er både billige og bra, noe som gir gode muligheter for video mens man flyr.

Petroleumsbransjen har allerede tatt i bruk droner og vi tror de vil brukes i enda større grad i fremtiden. Oppgaver som droner kan utføre for petroleumsbransjen er i hovedsak situasjoner der det er vanskelig for mennesker å komme til: inspeksjoner høyt oppe i en oljerigg, en fjellvegg som er vanskelig å nå til lands, oversikt ovenifra, med mer.

Dagens droner flys i hovedsak med visuell kontakt, der man er avhengig av å se dronen klart og tydelig for å kunne fly. På avstand blir det vanskelig å se hvilken retning dronen er vendt og dermed vanskelig å styre. Det finnes muligheter for live videostrøm, men disse gir ofte et flatt og lite bilde det er vanskelig å fly etter.

Disse problemene er det vi ønsker å løse.

## 1.2 Vår løsning

Vi ønsker å gi en mulighet til å manøvre droner på lang avstand med dårlig visuell kontakt. Løsningen vår skal oppnå dette ved å kombinere bildene fra to videokameraer på et Quadkopter med Virtual Reality-briller og styre retningen til kameraene ved hjelp av headtracking.

To videokameraer gjør at VR-brillene kan vise forskjellige bilde på hvert øye, og man oppnår dermed en ekte 3D-effekt og en mye bedre dybdeforståelse av hva man ser foran seg. VR-brillene som brukes er Oculus Rift. Disse brillene dekker hele synsfeltet med bilde og gjør at man får en altoppslukende, såkalt *immersive*, opplevelse av det man ser. Løsningen tar sikte mot at brukeren får en følelse av å faktisk henge under dronen med full oversikt over det som skjer.

Oculus Rift brillene har innebygd headtracking, så man på en PC kan lese av posisjonen til hodet til brukeren. Dette utnytter løsningen vår ved å rotere kameraene på Quadkopteret i to akser på samme måte som brukeren beveger

hodet. Å snu hodet til venstre og deretter få et bilde fra kameraer som er rotert til venstre vil gi en enda bedre kontroll over nøyaktig hva man ser, og en enda bedre følelse av at man er der det filmes.

Løsningen vår er i så måte ikke et konkret eksempel på hvordan petroleumsbransjen kan bruke droner til nye oppgaver enn de gjør i dag, men heller en ny styringsmekanisme som kan integreres i eksisterende og fremtidige oppgaver der droner brukes. Vi ser for oss at en viderutvikling av prototypen vil kunne gi et ferdig system man kan montere på hvilken som helst drone der man ønsker bedre kontroll i fremtiden.

# 2 Produktet

For å fullføre prosjektet var det behov for å finne løsninger på flere separate problemer. Det var lite overlapp mellom delproblemene, men det overordnede prosjektet var avhengig av at alt fungerer sammen. De følgende avsnittene forteller hva som måtte utføres i hver del av prosjektet.

## 2.1 Overordnet design

Figur 1: Overordnet design

Prosjektet kan deles inn i tre overordnede deler.

1. Anheng til quadkopter med servoer, sendere og kretskort

   Quadkopteranhenget lages slik at to kameraer kan roteres om to akser ved hjelp av servoer. Anhenget bærer også et kretskort med mikrokontroller til å styre kamera og servoer. Trådløs sender for video og mottaker for servo kobles til nevnte kretskort. Strømforsyning leveres av quadkopterets batterier.

2. Hodesporing fra Oculus Rift og oppdatering av kameraservoer

   Hodebevegelser registreres i Oculus Rift og retningskoordinatene sendes til PC. Disse koordinatene sendes trådløst til mikrokontrolleren på quadkopteranhenget som oppdaterer retningen kameraene peker ved å vri på servoene.

3. Videostrøm og visning i Oculus Rift

   To kamera sender video til radiosendere som videre sender signalet til mottakere koblet i en datamaskin. Datamaskinen omformer deretter videostrømmen til et bildeformat som kan vises i Oculus Rift.

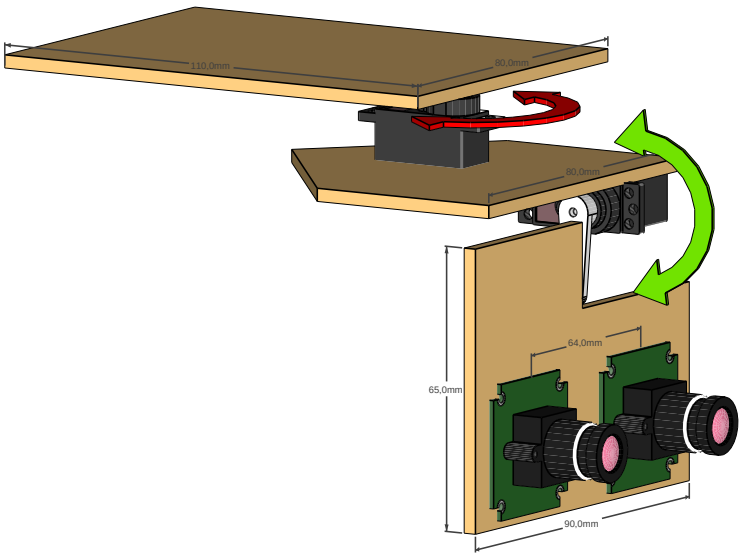
## 2.2 Anheng

For å kunne styre retningen kameraene peker, ønsket vi å kunne rotere dem om to akser. Vi har derfor bygget vårt eget anheng som kan sees i figur 2.

Dette anhenget består i hovedsak av tre finerplater og to servoer montert sammen. Den øverste platen er for montering i quadkopter. I den er det montert en servo som roterer en annen plate i y-retning. Denne platen har igjen en servo som roterer den tredje platen i x-retning. På den tredje platen er kameraene montert. Kameraene er montert med en avstand mellom sentrum av linsene på ca. 64 mm, da det er gjennomsnittlig *interpupillary distance*, avstand mellom pupillene, på voksne mennesker[1].

Anhenget monteres på quadkopteret, og må tilpasses hvilken type quadkopted man skal modifisere. Valget falt på *DJI Phantom 2*, da det har høye bein og dermed mye plass under, som kan sees i figur 1. Siden selve quadkopteret ikke er hovedfokus i vårt prosjekt var det også bra at det kommer ferdigmontert og klart til å fly.

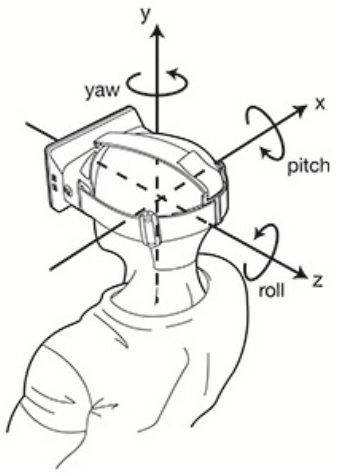
I vårt tilfelle monteres anhenget ved at bakstykket på den øverste platen skrues fast i gjenger på undersiden av quadkopteret. Den lange platen gjør at de roterende delene kommer langt nok ut fra beinene til at de ikke vikler seg inn i dem.




Figur 2: Anheng med servoer som roterer kameraene

## 2.3 Hodesporing

Oculus VR leverer utviklerverktøy til å lage innhold til Rift-platformen[1][2]. Ved hjelp av tilgjengelige funksjoner blant utviklerverktøyene kan man hente ut informasjon fra gyroskopet, akselerometeret og magnetometeret som følger med Oculus Rift. Informasjon fra disse sensorene brukes til å oppdatere orienteringen av bildet som vises i Oculus Rift, samt i vårt tilfelle oppdatere retningen kameraene peker. Figur 3 viser koordinatsystemet Oculus Rift innretter seg etter.




Figur 3: Oculus Rifts høyrehendte koordinatsystem. Ved å vende hodet oppover, vil sensorene registrere positiv endring om x-aksen. Å vri hodet mot venstre registreres som positiv endring om y-aksen og ved å vende hodet på skakke mot venstre registreres positiv endring om z-aksen. Rotasjoner om x-, y- og z-aksen betegnes som henholdsvis pitch, yaw og roll.

### 2.3.1 Orientering i Oculus Rift

Når brukeren av Oculus Rift beveger på hodet vil sensorene registrere dette og sende sensordata til brukerens datamaskin. Verdiene som mottas kan transformeres til et ønsket verdiområde. Verdiområdet vi ønsker å benytte vil være retninger som ligger innenfor servoenes utslagsområde. Servoene som benyttes i dette prosjektet kan rotere om 160 grader, noe som betyr at alle bevegelser innenfor et 160 grader utslagsfelt vil være tillatt. Hodebevegelser som registreres utenfor endeverdiene i verdiområdet settes til nærmeste endeverdi. Ligning 1 viser hvordan logikken vi benytter oss av setter verdiene slik

vi ønsker dem. $f$ tar retningskoordinater fra Oculus Rift og transformerer dem til et verdiområde som er innenfor servoenes rekkevidde.

$$f(x) = \begin{cases} 0 & : x < 0 \\ x & : 0 \leq x \leq 160 \\ 160 & : x > 160 \end{cases} \quad (1)$$

Siden vi benytter oss av to servoer vil vi kun kunne støtte rotasjoner om x- og y-aksen. Dette vil si at å vri hodet mot høyre og venstre og å vende hodet opp og ned, samt naturligvis kombinasjoner av disse vil være støttet. Det vil ikke være mulig å sette hodet på skakke (rotasjon om z-aksen) og forvente påfølgende oppdatering av bildet. En slik bevegelse vil føre til konfikt mellom hva øynene ser og hva balansesansen i øret registrer og kan fremkalle ubehag.

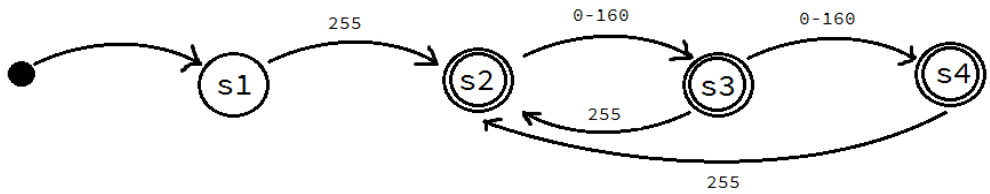
### 2.3.2 Dataoverføringsprotokoll for mikrokontroller

Mikrokontrolleren på quadkopteret er programmert til å forstå følgende dataoverføringsprotokoll: Starttegn etterfulgt av x-retning etterfulgt av y-retning. Protokollen består av tre elementer, hvor hvert element opptar 1 byte (8 bits). Starttegnet er satt til 255 ($2^8$ - 1 = $(11111111)_2$), en verdi som ikke er mulig å oppnå fra verdiområdet til servoene (0-160). Dette gjør at dersom starttegnet mottas av mikrokontrolleren, vil den forvente at de neste 8 bitene vil være x-retningen og de 8 deretter påfølgende bitene er y-retningen. Uansett hvor et starttegn mottas vil kontrolleren deretter forvente koordinater i rekkefølgen beskrevet. Figur 4 viser reglene for hvordan mikrokontrolleren skal tolke data den mottar.

### 2.3.3 Overføring av retningskoordinater fra datamaskin til quadkopter

Programmet vårt kjører en løkke som i hver iterasjon transformerer x- og y-koordinatene og setter opp pakkene som skal overføres i henhold til protokollen beskrevet i avsnittet over. I dette prosjektet benytter vi sendere for seriell dataoverføring (UART). Moderne bærbare datamaskiner mangler porter for dette formatet, slik at vi er avhengig av et *breakoutboard* for å kunne kommunisere med senderne våre. Et breakoutboard plugges til en USB-port i datamaskinen i ene enden, og til trådløs sender i andre enden. Når vi gjennom programmet sender data til den tilkoblede USB-porten, vil et breakoutboard

omforme signalet fra USB til UART-format og deretter videreformidle dette til senderen.



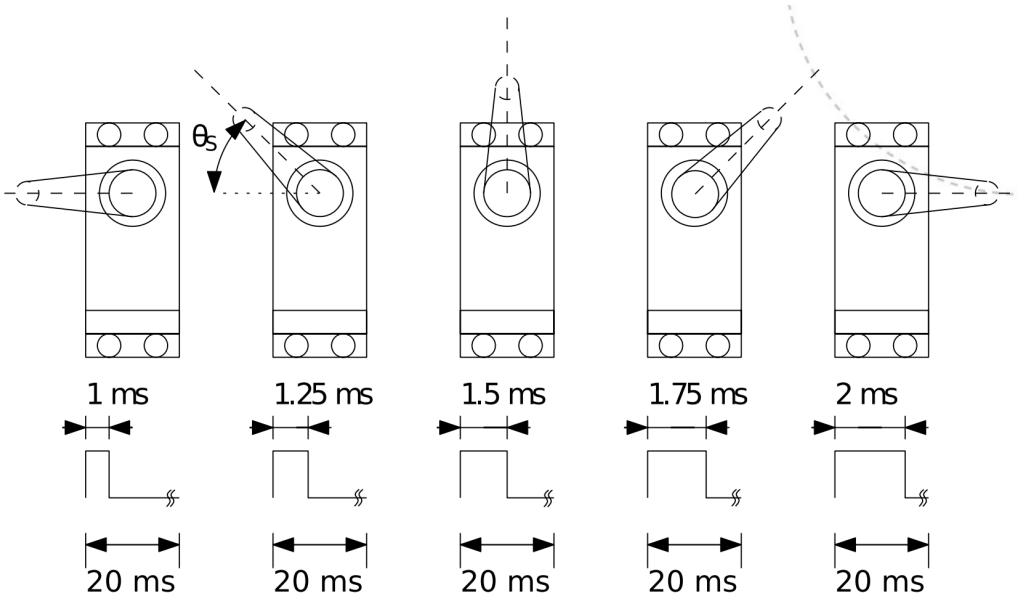

Figur 4: Signaltolkningen starter i tilstand s1, hvor mikrokontrolleren venter på starttegn. Ved tilstand s2 har mikrokontrolleren mottatt starttegn og venter på x-retningskoordinat. Ved s3 er x-koordinatet mottatt og retning til gjeldende servo oppdatert. Kontrolleren venter deretter på y-koordinatet. Den siste tilstanden er s4 og her har y-koordinatet blitt mottatt og servo oppdatert. I tilstand s3 og s4 er det mulig å falle tilbake til s2 dersom et starttegn mottas. Dersom ingenting mottas i løpet av 5 sekunder eller ugyldige verdier mottas starter man på nytt igjen i s1 (ikke inkludert i figur).

Mottakeren på quadkopteret tar i mot data som sendes, formidler dette til mikrokontrolleren som oppdaterer posisjonene til servoene ved å følge reglene i Figur 4.

### 2.3.4 Overføring av signaler fra mikrokontroller til servoer

Servoene styres via hver sitt PWM (puls bredde modulert) signal. Possisjonen til servoen vil være proposjonal med signalets duty cycle, altså hvor mye det er høyt i forhold til lavt, se Figur 5.

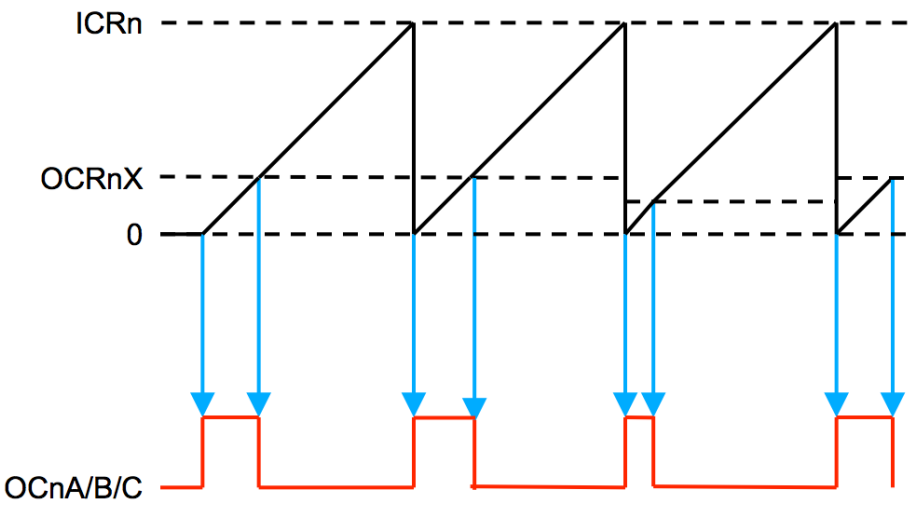
Mikrokontrolleren har som jobb å lage dette signalet, det gjør den ved hjelp av innebygde timere, som teller opp til når en utgang skal være høy eller lav, se Figur 6. Den har 2 timere for hver utgang, og når en timer har telt opp til en gitt verdi, setter den hennholdsvis en utgang høy eller lav. Ved å endre verdien på hva den skal telle opp til før man endrer verdien på utgangen avgjør duty cycelen.




Figur 5: Når duty cycle er 5% vil servoen stå på 0, mens når den nærmer seg 10% vil den stå på 160 grader.


### 2.3.5 Kretskort

For å koble hele systemet sammen måtte vi lage et kretskort. Denne skulle ha som oppgave å koble UART trancieveren, og servoene til mikrokontrolleren. Det ville også være behov for en strømforsyning, i form av en lineær regulator, som tok strøm fra quadkopterets batteri, og omformet dette til 5V. Kretskortet ble også utstyrt med LEDs for å kunne gi en statusindikator på dataoverføringen. For eksempel når en datapakke mottas, vil det lyse et grønt lys, og når kommunikasjonen med bakken oppører, vil det lyse et rødt lys. Kretskortet ble designet i CadSoft Eagle[7], etset og til slutt loddet på komponenter. Kretskortet ble programmert, testet og påmontert alt, resultatet er at kameraene følger bevegelsene til Oculus Rift.




Figur 6: OCnA/B/C er utgangssignalet som sendes til servoen. OCRnX og ICRn er registerene til timerene. Ved å øke verdien til OCRnX vil man øke duty cycelen, og dermed øke vinkelen på servoarmen.

## 2.4 Videolink

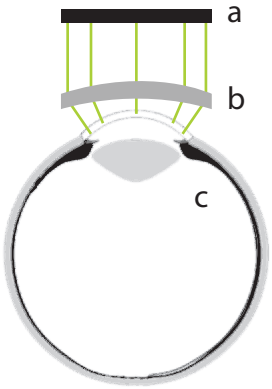
På quadkopteret er de to kameraene koblet i hver sin trådløse videosender, som igjen er koblet i batteriet for strøm. Disse senderne sender da videosignalet trådløst ned til hver sin mottaker på bakken. Sender/mottaker er standard hyllevare, allerede i bruk i mange droner. Mottakerne leverer videosignalet som RCA (kompositt) som man så kan plugge inn i en eller annen form for skjerm.

Ettersom vi har to videostrømmer vi ønsker å flette sammen og at Oculus Rift ikke er en hvilken som helst skjerm (mer om dette i neste seksjon), må vi behandle videobildene på en PC. Så neste steg er å digitalisere videostrømmen. Det gjør vi ved hjelp av to stykk *RCA til USB konverterere*, som lar oss koble det mottatte signalet fra mottakerne til en PC.

På PCen behandles så bildene i sanntid slik at de kan vises riktig på en Oculus Rift uten for mye forsinkelse. Mer om dette kommer etter neste seksjon.

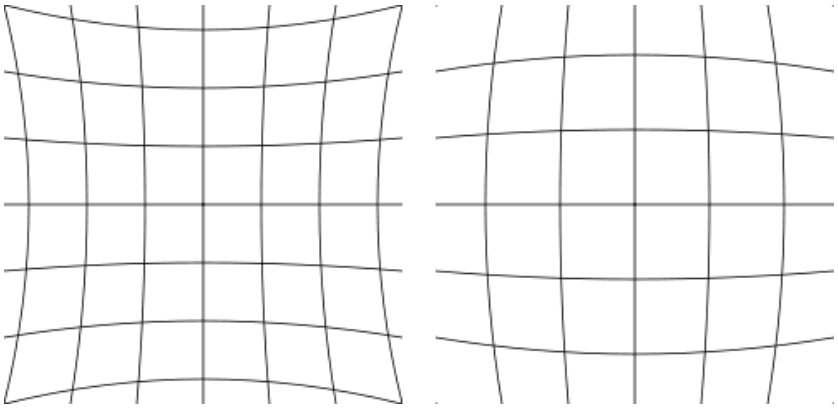
### 2.4.1 Bildeforvrengning

De som har sett Oculus Rift i aksjon før, har nok lagt merke til det rare bildet som vises på PC-skjermen. Figur 10 viser et slikt bilde. Om man derimot ser på samme bilde igjennom en Oculus Rift, ser det hele ut som det skal gjøre.




Figur 7: Linsene (b) i Oculus Rift får bildet (a) til å se ut som om det dekker hele synsfeltet til øyet (c)

Grunnen til dette er måten Oculus Rift virker på. For å dekke hele synsfeltet bruker den linser for å forvrenge et bilde fra en helt vanlig, flat, rektangulær skjerm. Figur 7 viser hvordan et bilde blir forvrengt av en linse til å dekke hele synsfeltet. Figur 8a viser måten linsen forvrenger bildet, en såkalt *pincushion* forvrengning. Om man viser et vanlig rektangulært bilde på en Oculus Rift vil det altså se forvrengt ut som i figur 8a, bare enda mer ekstremt.



Figur 8: a) pincushion til venstre b) barrel til høyre

Det motsatte av pincushion forvrengning er en *barrell* forvrengning, vist i 8b. Så for at bildet ikke skal se rart ut på Oculus Rift, gjør man en barrel forvrengning først for å oppheve effekten av å vri bildet rundt hele synsfeltet.

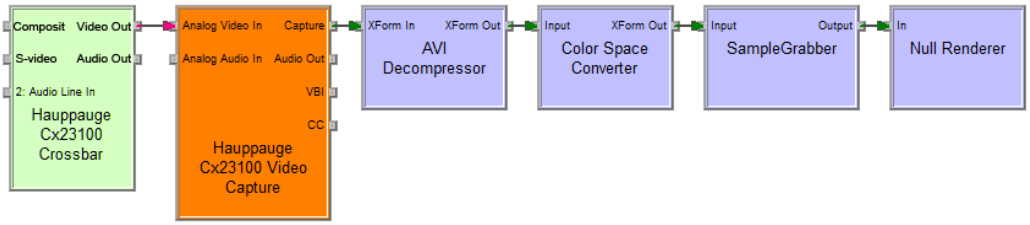
### 2.4.2 Behandling av videostrøm

Så vi må altså, i sanntid, forvrenge bildene i videostrømmen med barrel forvrengning før de blir vist på Oculus Rift.

Det finnes flere avspillere for å vise video på Oculus Rift, men ingen av dem kunne brukes. Årsakene til dette var flere, men i hovedsak at spillerne tok for seg video*filer*, ikke sanntidsvideo, og kunne heller ikke koble sammen to videostrømmer. Løsningen ble å skrive vår egen programvare. Ettersom den medfølgende programvaren til RCA til USB konverterne hadde en forsinkelse på nesten to sekunder, kunne vi ikke bygge programmet vårt oppå dette.

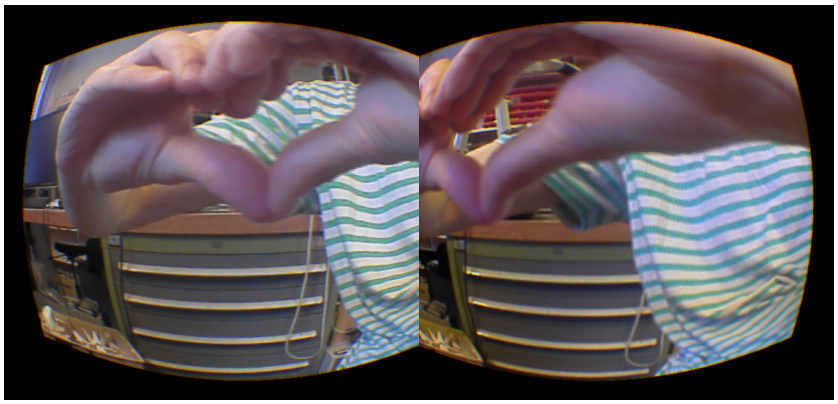
Programmet vårt snakker istedet direkte med *driveren* til konverterne og henter ut rå-dataen disse får inn. Det vi gjorde var å sette opp en *DirectShow*[3] graf, der vi sender rå-dataen igjennom flere steg for å få frem et

bilde (*frame*) vi kan bruke. For å gjøre dette måtte vi bruke C++ og lavnivå COM[4]. Grafen som er satt opp er vist i figur 9. Den grønne *Crossbaren* er selve den fysiske enheten, som er koblet opp mot den oransje *Video Capture* driveren. Det den gir ut sender vi så igjennom en *Decompressor* og *Color Converter* for å få et bilde. Dette sendes videre til en *SampleGrabber*[5], som bare er et buffer der det seneste bildet i videoen ligger slik at vi kan hente det ut og behandle det. Siden vi må gjøre en del prosesseringer på bildet selv, skal det ikke vises og bildet sendes derfor til en såkalt *Null Renderer* som bare forkaster bildet.




Figur 9: DirectShow graf for videostrøm

En slik graf bygges for hver videostrøm og deretter settes det opp en loop som henter ut det seneste bildet fra SampleGrabberen for hver video. Hvert slikt bilde blir så forvrengt med en barrel forvrengning før det blir vist på hver sin side av skjermen slik at det kommer på hvert sitt øye. For å forvrenge bildet effektivt gjøres det med en *shader*[6] på skjermkortet (GPU). Resultatet er et bilde som i figur 10.



Figur 10: Bilde fra hvert kamera med forvrengning, klart for å vises på en Oculus Rift

# 3 Resultater og diskusjon

Prosjektets mål ble nådd før innleveringsfristen for prosjektrapporten. Den siste landsbydagen fikk to av gruppemedlemmene prøvd ut systemet slik det skal brukes, og resultatet var tilfredsstillende. Det er likevel flere områder designet av prototypen vår kan forbedres. Det ble oppdaget at millimeterpresisjon i monteringen av kameraene er essensielt for å kunne levere troverdige bilder i Oculus Rift. Noen få millimeter med unøyaktighet i retningen de to kameraene peker vil gi vesentlige utslag i bildet. Når kameraene er riktig montert vil hjernen sette sammen bildene som blir servert til hvert øye på riktig måte, og når det er små unøyaktigheter vil hjernen forsøke å kompensere, noe som blir svært utmattende og plagsomt etterhvert. Når bildene ikke er sammenstilt vil dybde i bildet være fraværende, noe som var ett av hovedmålene å oppnå i dette prosjektet. Det skal nevnes at etter testflyvningen fikk vi gjort rede for hvordan kameraene skal justeres, og testet dette ut på bakken, men altså ikke en testflyvning med fullverdig immersjon. Dette forventes å bli rettet opp dagene før presentasjonen den 24. april.

Når det gjelder anhenget er det flere svakheter som kan føre til havari dersom en skulle være uheldig. I Figur 2 kan det på servoen skimtes en skrue som fester det hvite roterbare plastelementet til servokassen. For begge servoene er det denne skruen som bærer all vekten til de påfølgende delene av anhenget. Det er også disse leddene som må tåle akselerasjonen når quadkopteret flyr og vender på seg i luften. Det ble forsøkt å montere et mest mulig redundant sikringssystem i anhenget, med det formål at vekt fordeles og for også å redusere antallet kritiske punkter. I servoenes tilfelle var det ikke mulig å utbedre dette, og vi er foreløpig på dets nåde når quadkopteret er ute og flyr. Det ble benyttet finérplater som materiale i anhenget, og dette løper alltid den risiko å begynne å sprekke dersom man skrur og sager i dem. Det var gjennom prosjektperioden planlagt å benytte 3D-printeren ved Institutt for petroleumsteknologi og anvendt geofysikk til å fremstille anhenget, men forsinkelser i leveringen av plasten som printeren benytter gjorde at vi måtte bytte til trevirke. Dersom vi hadde fått mulighet til å 3D-skrive delene til anhenget, ville det være enkelt å gjenskape produktet vårt slik vi presenterer det. Det er likevel forskjellige styrker mellom det å bruke trevirke og 3D-plast. Ved bruk av tre kan man sage ut deler til anhenget og deretter går det hurtig å modifisere dem dersom noe ikke passer. Til gjengjeld vil det kreve mye mer tid å få testet ut prototyper dersom det blir benyttet en 3D-skriver, siden skriveren trenger tidsmengder på størrelse med en arbeidsdag til å ferdigstille modellen man laster inn. Samtidig er det slik at dersom man har fått på plass en fungerende modell til 3D-skriveren vil man kunne enkelt

reprodusere delene. Man kan forestille seg at man kan starte 3D-skriveren på kveldstid dagen i forveien og deretter hente de ferdige delene morgenen etter. Med både plast og trevirke må man skru og file for å montere delene sammen, så det er ikke et matriale som kommer bedre ut enn det andre når det angår ferdigstilling. Det skal nevnes at de aller fleste har mest erfaring med å jobbe med tre som byggematriale sammenlignet med plast til bruk i samme formål. Det skal likevel ikke være særlig utfordrende å lære seg grensene for saging og skruing i 3D-plast og dets bruksområder, så denne fordelen i favør trevirke er neglisjerbar.

Skjermoppløsningen i utviklerutgaven av Oculus Rift som vi besitter leverer 640x800 piksler på hvert øye. Dette er merkbart kornete og hver piksel kan skilles fra hverandre. Dette har den konsekvens at bildet som vises vil virke uklart. Dette er en effekt som merkes uansett applikasjon man kjører i Oculus Rift. Oculus VR hevder at dette vil bli forbedret i neste utviklerutgave samt forbrukerversjonen av Oculus Rift. Når bildet er kornete vil hjernen automatisk be øynene om å fokusere ytterligere for å oppheve uklarhetene. Dette er arbeid som gjøres forgjeves, siden bildekvaliteten er prisgitt skjermoppløsningen, og brukeren blir sliten i både øyne og hjerne. Vi forestiller oss at det ikke kommer til å være mye arbeid i å tilpasse vårt produkt til nyere versjoner av VR-briller, slik at opplevelsen i Oculus FPV vil bli enda bedre. Oppløsningen i kameraene vi benytter er omtrentlig sammenstilt med oppløsningen som Oculus Rift tilbyr, men det vil likevel være mye å hente med større oppløsning i 3D-brillene.

Det ble ikke registrert store forsinkelser i overføring av videostrøm fra og servoposisjon til quadkopteret. Med tanke på respons var løsningen vår slik den forligger tilfredstillende. Når det gjelder oppdateringsfrekvens på servoposisjoner, var ikke dette så hurtig som det er mulig å få det, men det var likevel mer enn tilstrekkelig til å gi en naturlig opplevelse. Servoposisjonene oppdateres 30 ganger i sekundet, og med litt ekstra arbeid kan dette forbedres.

For å kunne forsyne servoene, kameraene og senderne med strøm, ble det i vårt tilfelle nødvedig å lodde seg på quadkopterets strømforsyning. Dette er et steg opp i nødvendige forkunnskaper for å kunne reprodusere produktet vårt. Det ideelle hadde vært å unngå at nye brukere må lodde dette samt å måtte lage sitt eget kretskort. Kunnskapen i gruppen var tilstrekkelig til at vi kunne produsere dette på egenhånd, men en bedre løsning ville vært om mest mulig av delene våre kunne bli ferdigstilt uten å måtte ha tilgang til et el-verksted for å kunne lodde. Alternativer til å måtte lodde sitt eget kretskort er å montere en Arduino eller tilsvarende. Med en slik platform skal det være enkelt å styre servoene.

# 4 Fremtid for produktet

Det ideelle målet med prosjektet er å kunne fremstille produktet slik at det er mest mulig plug-and-play. Dette vil si at det ikke skal være store utfordringer knyttet montering av anhenget og oppsettet av sendere og programvare. Mye er avhenging av hvilket quadkopter som blir benyttet, hvor anhenget må tilpasses til forskjellige modeller. Med en fungerende modell til 3D-printing vil det være enkelt å gjenskape delene vi har benyttet, og dette vil gjøre prosjektet ytterligere reproduserbart.

Produktet vårt kan ha mulighet til å bære andre typer kamera og flere sensorer, slik at man kan danne seg et mer komplett bilde av situsjonen man er satt ovenfor. Med mulighet til å montere IR-kamera vil det være mulighet til å registrere varmeutvikling, med gassensorer kan lekkasjer detekteres. Flere av disse ekstra funksjonalitetene vil allerede være støttet av eksisterende løsninger, og for gasslekasjer og varmeutvikling vil dette være hendelser som mest sannsynlig vil bli plukket opp av statiske og integrerte sensorer. Hovedstyrken for vårt produkt vil være i integreringen av VR-briller som omfatter et fullstendig synsfelt, og muligheten til å snu hodet mot det som vekker brukerens interesse og deretter få bildet oppdatert. Bruken av produktet kan være nyttig i situsjoner hvor det er behov for å inspisere ting i krevende og farlige omgivelser, hvor alternativet ville vært kostbart, tidkrevende og arbeidsintensivt. Eksempler på disse er utredning av utilgjengelig terreng, eksempelvis i forbindelse med rassikring. I slike situasjoner kan det være for farlig å benytte mennesker til dette arbeidet. Andre bruksområder er i forbindelse med inspeksjon av deler av bygninger som er vanskelig å nå på konvensjonelt vis. Eksempler her er bærestag på oljeplattformer, kjøletårn og høybygg for nevne flere.

Det er fullt mulig å holde seg til bruk av ett kamera i oppsettet vårt. Dette gjør det mulig å kombinere produktet vårt med andre prosjekter i landsbyen som har kombinert kamera med droner.

# 5 Konklusjon

Det ble i dette prosjektet gjort mulig å benytte Oculus Rift til å vise video fra et quadkopter, samt styre retningen kameraene peker ved hjelp av hodesporing i Oculus Rift. Som det er skrevet i delene over så er det mye som kan forbedres for å gjøre produktet bedre og enklere å reprodusere, men vi har vist at det er mulig med hyllevare å lage en fungerende løsning. Det skal merkes at det har vært behov for å kunne lodde og ha tilgang til verksted, noe som ikke alle har mulighet til.

Vi fikk til det vi ønsket i prosjektet, og sluttresultatet viser at produktet vårt gir god kontroll og oversikt når man flyr. Vi kan derfor anbefale videre testing og utvikling av vårt system.

# Referanser

[1] Richard Yao, Tom Heath, Aaron Davies, Tom Forsyth, Nate Mitchell, Perry Hoberman *Oculus Best Practices* Oculus VR, Inc. 2014.

[2] Michael Antonov, Nate Mitchell, Andrew Reisse, Lee Cooper, Steve LaValle, Max Katsev *Oculus SDK Overview* Oculus VR, Inc. 2013.

[3] *Microsoft DirectShow application programming interface* `http://msdn.microsoft.com/en-us/library/windows/desktop/dd375454%28v=vs.85%29.aspx` Microsoft, 2014.

[4] *DirectShow, COM and C++* `http://msdn.microsoft.com/en-us/library/windows/desktop/dd390351%28v=vs.85%29.aspx` Microsoft, 2014.

[5] *SampleGrabber* `http://msdn.microsoft.com/en-us/library/windows/desktop/dd377544%28v=vs.85%29.aspx` Microsoft, 2014.

[6] *Understanding the Oculus Shader* `http://rifty-business.blogspot.com/2013/08/understanding-oculus-rift-distortion.html` Brad Davis, Rifty Business, 2013.

[7] *CadSoft Eagle* `http://www.cadsoftusa.com/` CadSoft USA, 2014

# 6 Vedlegg

## 6.1 Produkter

Tabell 1 viser de viktigste komponentene brukt i produktet vårt.

| *Type* | *Komponent* | *Antall* |
|---|---|---|
| Quadcopter | DJI Phantom 2 | 1 |
| Kamera | CMOS Camera Module - 728x488 SEN-11745 | 2 |
| Videolink | 5.8GHz FPV AV 600mW RC832 | 2 |
| Videokonverter | Hauppauge USB-LIVE 2 | 2 |
| VR-briller | Oculus Rift Developer Kit | 1 |
| Servo | Servo Sub-Micro ROB-09065 | 2 |
| USB til serial | FT232RL Breakout BOB-00718 | 1 |
| Seriallink | Long Range 433MHz UM96 WRL-00155 | 2 |
| PC | Vilkårlig, må kjøre Windows | 1 |

Tabell 1: Produkter

## 6.2 Kildekode

Kildekoden til mikrokontrolleren og programmet på PCen finnes tilgjengelig i et Git repo på følgende URL: `https://github.com/Matsemann/oculus-fpv`

Instruksjoner for bygging av kildekode finnes i filen *README.md* i repoet.