

Utvikling av webbasert Augmented Reality

Høstsemesteret 2021

Bacheloroppgave: HPR/DM-016

Kandidatnummer: 401

Utført av: Caroline Berg

Utgavedato: 4.desember 2021



Universitetet i Agder, 2020

Fakultet for teknologi og realfag

Jon Lilletuns vei 9, 4879 GRIMSTAD

Tlf. 37 25 30 00

Forord

Denne rapporten omhandler resultatet av et bachelorprosjekt innen multimedieteknologi ved Universitetet i Agder høsten 2021. Prosjektet er en del av DAT303 og gir deltagende studenter 20 av bachelorstudiets totale 180 studiepoeng.

Takk rettes til førstelektor Morgan Konnestad og overingeniør Jostein Nordengen ved Universitetet i Agder i Grimstad for veiledning og oppfølging gjennom prosjektetperioden.

Studenten ansvarlig for gjennomføringen av prosjektet er Caroline Berg
31. mai 2024

Sammendrag

Dette prosjektet har hatt som formål å utforske forskjellige metoder og løsninger for en webbasert applikasjon med Augmented Reality-funksjonalitet. Heroku er brukt som plattform for web-applikasjonen og er satt til å bygge et repository fra GitHub. Three.js er benyttet til å forme et visuelt 3D rom samt kontrollere kameranavigasjon i 3D-rommet. For å hente sensordata fra enhetens orientasjon og forflytning er web-APIene *deviceorientation* og *devicemotion* benyttet. Til bildebehandling og tracking benyttes *OpenCV.js*-biblioteket hvor da *CannyEdge* og *HarrisCorner* er tatt i bruk. Rapportens resultat viser en enkel web-applikasjon som er laget med formål om å benyttes som en testapplikasjon. Resultatet tar også for seg målinger gjort med sensordata og hvordan dette kan brukes videre i en Augmented Reality-applikasjon. Et enkelt eksempel på interaktivitet i 3D er også presentert.

Nøkkelord: Augmented Reality, Three.js, OpenCV.js, bildebehandling og tracking, sensor, Web API, Heroku

Link til online test-applikasjon med base på heroku: open-ar.herokuapp.com[1].

Figurer

1	Forskjellene mellom AR, VR og MR illustrert av AR VR tech[2]	2
2	Applikasjonsutviklingen foregikk lokalt før den publisert på GitHub	5
3	Applikasjonsstrukturen som viser hvor de ulike modulene sender data.	6
4	Mixed Reality definert som spekteret av teknologi mellom det digitale og virkeligheten. Illustrasjonen er hentet fra Microsoft[3]	9
5	Illustrasjon av Clint Ford for artikkelen "How Does Augmented Reality Work?"[4]	11
6	Modell laget av Microsoft som beskriver hva skytjenestene vanligvis inneholder	13
7	Bevegelsessensorer brukt til orientering i VR-teknologi[5]	16
8	Smarttelefonens koordinatsystem hentet fra Android Developers[6]	16
9	Koordinatsystemet til kamera-objektet i Three-scenen og smarttelefonen.	17
10	Three.js øverst på 6 av 7 vurderinger av rammeverk for 3D i Javascript	18
11	Ansikt gjenkjenning. Illustrasjon fra Huawei Developers for Medium.com[7]	19
12	QR-kode markør som erstattes med en levende modell. Bilde:digit.in[8]	19
13	Eksempel på bilde som en tar Edge Detection på	20
14	Edge Detection resultat	20
15	Figur som beskriver dybdemåling ved stereosyn [9]	21
16	Sensor API innholder funksjoner som henter informasjon fra sensorene.	23
17	En "EventListener" som lytter til endringen i orienteringen til enheten.	23
18	Flytskjema for å spørre om tilgang til sensorene ved bruk av permission API.	24
19	Akselerasjonsmåling som en funksjon av tiden delt opp per tidsenhet (dt)	25
20	Oppbyggning av handleMotion.js som henter data fra akselerometeret og regner ut farten og strekningen.	27
21	Eksempladata for kontroll av formlene	28
22	Enheten ble flyttet 30cm langs x-aksen.	28
23	Grafene fra eksempladata som ble brukt til å kontrollere formlene som blir brukt i utregningen av hastighet og strekning.	28
24	Ved å rotere enheten ser vi at gravitasjonen påvirker enheten positivt oppover på grunn av en filtrert akselerasjon	29
25	Akselerasjonsdata fra test i 30 sekunder uten filtere.	29
26	Hastigheten over tid fra test i 30 sekunder uten filtere.	30
27	Strekningen over tid fra test i 30 sekunder uten filtere.	30
28	Akselerasjonsdata over tid med ZeroCounter	31
29	Hastighet og strekning ut i fra data i figur 28	31
30	Hastighet og strekning med påvirkning fra ZeroCounter	31
31	En scene med en 3D-modell og et perspektivisk kamera.	32
32	Struktur på bildebehandling	33
33	Bildet av et hjørne i farger	33
34	Bildet etter det er kjørt igjennom <i>CannyEdge</i>	33
35	Bildet etter det er kjørt igjennom <i>HarrisCorner</i>	33
36	3D skisse av prosjeksjon av detektert hjørner inn på bildeplanet	34
37	Skisse av 2 punkter som er projektert i bildeplanet	35
38	Skjermbilde av applikasjonen under oppstart i localhost.	36
39	Resultatet av en akselerasjonsmåling med tilpassede filtere	38
40	Three-scenen med gjennomsiktig 3D mesh (lilla) og en grønn 3D boks	39
41	Kamera sitt bilde i gråskala bilde	40
42	Gråskala bildet kjørt igjennom <i>CannyEdge</i>	40
43	<i>CannyEdge</i> bildet kjørt igjennom <i>HarrisCorner</i>	40
44	<i>HarrisCorner</i> bildet kjørt igjennom <i>erode</i> og <i>dilate</i> filter	40
45	Bildet av et rom hvor 4 hjørner er valgt og tracket	41

Forkortelser

Benevning	Forklaring
AR	Augmented Reality
XR	Extended Reality
VR	Virtual Reality
MR	Mixed Reality
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
MEMS	Micro Electro Mechanical Systems
v	Hastighet [m/s]
v_{prev}	Hastighet ved sist måling
a	Akselerasjon [m/s^2]
dt	Tidsendring [s]
s	Posisjon [m]
s_{prev}	Posisjon ved sist måling
a_{prev}	Akselerasjon ved sist måling
d	Disparitet
f	Fokallengde / Brennvidde
Z	Avstand fra kamera til objekt
B	Avstand mellom kamera 1 og 2
x	Pikselavstand i bildeplanet
dir_a	Akselerasjonens retning [$+/- 1$]
dir_v	Hastighetens retning [$+/- 1$]
s_c	Kamerasensor størrelseforhold [$m/piksel$]

Innhold

Forord	i
Sammendrag	ii
Figurer	ii
Forkortelser	ii
1 Innledning	1
1.1 Motivasjon for oppgaven	1
1.2 Problemdefinisjon	3
1.3 Forutsetninger og begrensninger	4
1.4 Litteraturstudie	4
1.5 Problemløsning	5
1.5.1 Applikasjonsutvikling	5
1.5.2 Applikasjonsstruktur	6
1.6 Prosjektplan	7
1.7 Møtereferater	8
1.8 Timeliste	8
1.9 Rapportstruktur	8
2 Teori og metoder	9
2.1 Extended Reality	9
2.2 Augmented Reality	10
2.3 Hardware	12
2.4 Applikasjonsutvikling	12
2.4.1 Mobilapplikasjon	12
2.4.2 Web applikasjon	12
2.4.3 Skytjenester	13
2.5 Navigasjon og bevegelse	14
2.5.1 Knapper	14
2.5.2 Berøring	14
2.5.3 Sensorer	14
2.5.4 Koordinatssystemer og akser	16
2.5.5 Euler translasjon	17
2.6 3D-biblioteker for Javascript	18
2.6.1 Three.js	18
2.7 Bildebehandling og tracking	19
2.7.1 Edge Detection	20
2.7.2 Dybdeberegning	21

3 Løsningsmetode	22
3.1 Krav til løsningen	22
3.2 Webapplikasjon	23
3.2.1 Heroku	23
3.3 Navigasjon og bevegelse	23
3.3.1 Magnetometer	25
3.3.2 Akselerometer	25
3.3.3 Test av akselerometeret til avstandsberegning	28
3.4 3D-grafikk	32
3.4.1 Three.js	32
3.4.2 Sensorsdata i 3D-scenen	32
3.5 Bildebehandling	33
3.5.1 Dybdeberegning	34
4 Resultat	36
4.1 Applikasjonsdesign	36
4.2 Navigasjon og bevegelse	37
4.2.1 Magnetometer	37
4.3 Akselerometer	38
4.3.1 Andre bruksområder	38
4.4 3D-grafikk	39
4.5 Bildebehandling	40
4.5.1 Dybdeberegning	41
5 Diskusjon	42
6 Konklusjon	44
Referanser	45
A Prosjektplan Våren 2021	1
B Prosjektplan Høsten 2021	2
C Møtereferater	3
D Timeliste	9
E Forprosjektrapport	10

1 Innledning

De siste årene har teknologi som “utvider virkeligheten” - Extended Reality(XR), vokst og selges til vanlige forbrukere i større grad. Målet for prosjektet er en teknisk applikasjon som skal teste muligheten for å oppnå enkle opplevelser ved bruk av Augmented Reality på enheter som ikke er kompatible med dagens applikasjoner.

1.1 Motivasjon for oppgaven

De siste årene har det vært en økt personlig eksponering av Virtual Reality-programvare. Virtual Reality er tilgjengelig for forbrukerne gjennom vanlige leverandører digitale og teknologiske enheter. Headsettene kan kobles til både spillkonsoller, smarttelefoner via applikasjoner eller kjøres separat. Store spillutviklingsplattformer som Unity[10] og UnrealEngine[11] tilbyr, sammen med mange andre, muligheter til å utvikle egne applikasjoner med noe enkle steg. Augmented Reality har også fått feste og er tilgjengelig i applikasjoner som Snapchat, PokemonGo eller andre kamera-applikasjoner.

I tillegg til økt eksponering av XR-teknologi har det blitt arbeidet med 3D-modellering og animasjon i studiet og i praksisperiode. Denne eksponeringen har ført til en styrket interesse for å se på bruksmulighetene for å 3D interaktivt i applikasjoner. I XR er 3D en essensiell grafikk for applikasjonene. I Virtual Reality skal brukeren flyttes inn i en virtuell virkelighet og stenger ute virkeligheten ved å påvirke flere av sansene. Augmented Reality(AR) ligger nærmere virkeligheten og plasserer 3D-grafikken inn omgivelsene brukeren befinner seg i. Flere leverandører av møbel og interiør, som VikingBad AS[12] og Compusoft Group[13] har tatt i bruk 3D. Bedriftene har egne 3D-rom hvor man kan få et realistisk bilde av rommet med interiør før man kjøper dem. Kunden kan bevege seg rundt i rommet ved å benytte VR-briller eller ved tastetrykk rundt i rommet. IKEA Place[14] er en AR-applikasjon som gjør det mulig å plassere møblene på gulvet i rommet man befinner seg i. Møblene skaleres etter rommets størrelse med 98% nøyaktighet og ved hjelp av AR-teknologien i AR Kit vil den få realistiske tekstiler med pålagt lys og skygge etter rommets belysning. Det er IKEA Place og lignende navigasjons applikasjoner, samt manglende tilgang på VR-headsett som gjorde at valget falt på Augmented Reality.

Etter testing med å lage enkle AR-applikasjoner i Unity[10] ble det oppdaget at det var vanskelig å videreutvikle AR-funksjonene med egne funksjoner ettersom de ligger inne i AR Core[15] og AR Kit[16] sine programvareutviklingsett. Samtidig ble kompatibilitet et problem da AR Core og AR Kit krever noen sensorer, operativsystemer eller prosessorer som ikke var tilgjengelig på smarttelefonene som ble forsøkt tatt i bruk. Dette begrenser bruk av applikasjoner utviklet med AR Core og AR Kit til forbrukere med de aller nyeste enhetene. At applikasjonene krever at forbrukerne har de nyeste modellene gjør det lite hensiktsmessig å promotere eller ta i bruk applikasjonen for bedrifter som driver med blant annet salg av interiør da det ikke kan garanteres at kundegruppen har de nødvendige operativsystemene eller enhetene som kreves. Dette vil mest sannsynlig endre seg i løpet av noen år da de eldste modellene fases ut og nye modeller vil ha nødvendige sensorer ved hjelp av nanoteknologiens utvikling[17].



Figur 1: Forskjellene mellom AR, VR og MR illustrert av AR VR tech[2]

Gjennom å erfare både komatibiltetsproblemer og utviklingbegrensning ved bruk av AR Core, ble fokuset endret. Det ble interessant å utforske de helt grunnleggende funksjonene og prinsippene i en Augmented Reality-applikasjon. Hva trenger de for å fungere? Kan de kjøres uavhengig av operativsystemene? Ved å kjenne til det grunnleggende ved AR-funksjonalitet vil det gi et grunnlag og nye muligheter for å enklere utvikle applikasjoner senere. Både med og uten ferdig programvare.

I løpet av bachelorstudiet Multimedieteknologi og -design, på Universitet i Agder i Grimstad, har fokuset vært at studentene skal ha kunnskap til å kunne bygge og produsere grunnleggende produkter ved bruk av tilgjengelige hjelpemidler og biblioteker. Ved å bruke grunnleggende kunnskap om applikasjonsutvikling og utforske oppbyggningen av Augmented Reality - ble oppgaven å teste mulighetene for å benytte innebygde og tilgjengelige sensorer til å utføre enkle AR funksjoner som er tilgjengelige for smarttelefoner uavhengig av operativsystem.

1.2 Problemdefinisjon

Målet med prosjektet er å lage en applikasjon med funksjonalitet som finnes i Augmented Reality uten å møte større kompatibilitetsproblemer. Den skal bygges opp ved hjelp av grunnleggende javascript og åpne biblioteker for å oppnå en større forståelse for hvordan Augmented Reality henter, analyserer og benytter data fra enhetens sensorer til å skape en sammenheng mellom det virtuelle og virkeligheten.

Ut ifra definisjonen av Augmented Reality (se kapittel 2.2) kan vi sette følgende krav til applikasjonen:

- Applikasjonen kombinerer virtuelle elementer og virkelig data.
- Applikasjonen har interaktive elementer som påvirkes av analyse i sanntid.
- Funksjonene påvirker elementer i et 3D-miljø.

Øvrige krav til applikasjonen:

- Materielle rettigheter skal respekteres.
- Programvare må ikke være lisensensert på en måte som hindrer offentliggjøring, videre distribusjon eller videreutvikling av applikasjonen.
- Applikasjonen skal være tilgjengelig for et flertall av enheter uavhengig av operativsystem.

De definerende kravene til AR-applikasjonene innebærer at applikasjonen skal inneholde en virtuell 3D-verden med mulighet for å legge til modeller. 3D-modellene skal kunne vises på skjermen samtidig som omgivelsene fra virkeligheten. For at de skal implementeres i virkeligheten må objektene ha et visst nivå av orientering eller bevegelse basert på analyse av omgivelsene. Det må foregå en informasjonsflyt mellom analyserende moduler og virtuelle objekter for tilpasning av de virtuelle objektene basert på analyse og mottatt data i sanntid. Denne informasjonsflyten skal gjøre det mulig oppdatere objektenes orientering eller posisjon i applikasjonen basert på brukeren bevegelser eller sensorenes registreringer.

Resultatet av prosjektet skal fokusere på om de ulike metodene og sensorene som testes til å utvikle AR-funksjonalitet oppnår et tilfredstillende resultat og kan brukes til å utvikle en applikasjon som kan distribueres.

Prosjektets målsetning ble satt i samråd med veiledere og har sammenheng med eksisterende teknologi og teknologiske muligheter.

1.3 Forutsetninger og begrensninger

Gjennomføringen av prosjektet forutsetter tilgang til PC, en smarttelefon med kamera og de vanligste bevegelsessensorene, samt tilgang til OpenSource-biblioteker med informasjon om sensorene. Under testing og utbedring av bildebehandling forutsettes det et godt opplyst rom, men synlige hjørner for best mulig målinger.

Omfanget av prosjektet begrenser seg til å kunne kjøre applikasjonen i Google Chrome med en smarttelefon med et operativsystem fra Android. Applikasjonen skal bidra til å vise prinsippene ved bruk av sensorer og kamera. Det er derfor ikke lagt vekt på design eller avanserte 3D-modeller i prosjektet. Projektet baseres kun på OpenSource-biblioteker for å ikke begrense muligheten til å kunne videreutvikle konseptet etter prosjektets slutt og samtidig slippe å tenke på rettigheter underveis.

Oppgavens omfang begrenses også av at det er lite kunnskap om temaet på Universitetet. Det er heller ikke forventet av en bachelorstudent på Multimedieteknologi og -design å skulle ha forkunnskap om Virtual Reality eller Augmented Reality da det ikke undervises før 3.semester på et eventuell masterstudie.

Det blir ikke lagt vekt på utseende da applikasjonen begrenser seg til å fungere som en testapplikasjon for å teste AR-funksjonalitet basert på grunnleggende javascript, OpenSource-biblioteker og tilgjengelige sensorer.

1.4 Litteraturstudie

Prosjektet baseres på Javascript som programmeringsspråk da det er det som har blitt brukt i tidligere semestere på bachelorstudiet. W3Schools[18] er et online bibliotek med veiledning med fokus på læring, testing og trening med blant annet Javascript syntaks. Her hentes det informasjon om grunnleggende syntaks og inspirasjon til løsninger.

For implementering av 3D-objekter ble dokumentasjonen til Three.js[19] benyttet. Three.js er et OpenSource javascript-bibliotek med god dokumentasjon og mange nyttige eksempler.

For å få bedre forståelse av AR-funksjonalitet og hva AR-applikasjoner benytter av data for å fungere, ble det laget enkle applikasjoner ved hjelp AR Foundation i Unity[20]. Det ble også benyttet enkle tutorials som er lett tilgjengelig på youtube ved blant annet søkeordene “AR Unity3d”[21].

For informasjon og dokumentasjon om bildebehandling, blant annet Edge Detection ved hjelp av kamera, ble sidene til OpenCV brukt[22].

Android Developer sin dokumentasjon[6] og Mozilla Developer[23] sine Web API's ble brukt i forbindelse med informasjon, implementering og uthenting av data fra enhetens sensorer.

1.5 Problemløsning

For å bli kjent med funksjoner som er sentrale for Augmented Reality-applikasjoner er det nyttig å teste ut og se på eksisterende applikasjoner. Ut i fra dette er det mulig å få en liten oversikt over sentrale bruksområder og forventninger til funksjonalitet.

Spillmotorer og utviklingsverktøy som Unity[10] og Unreal Engine[11] benytter seg av både OpenSource og lisensert programvare og har selv begrensinger på hva du kan benytte i applikasjonen. Det er allikevel en fin erfaring å ha med videre i arbeidet med AR-applikasjoner å ha jobbet med profesjonelt utarbeidet programvare laget til bruk i Augmented Reality. Dette er programvare som AR Core[15] og AR Kit[16]. Ved å lage enkle applikasjoner i Unity eller UnrealEngine er det mulig å bli kjent med oppbygging, begrensninger, forutsetninger og viktige komponenter for tilsvarende applikasjoner.

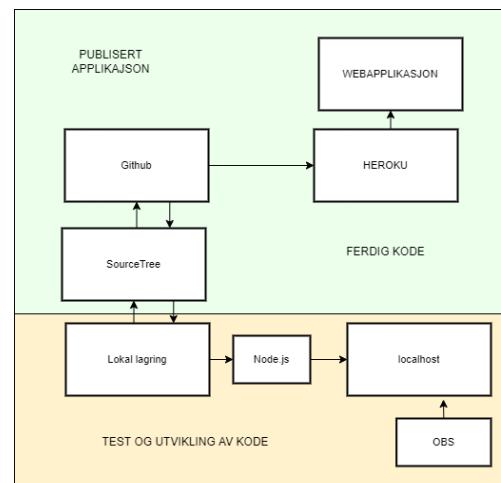
Før det startes på selve applikasjonen bør struktur og innhold planlegges. Det må lages en oversikt over ønsket funksjonalitet, moduler, design og eventuelle verktøy. Denne kan gjøre det enklere å holde seg til planen og få resultatene man er ute etter. Ut ifra produktoversikten kan det lages en prioriteringsliste og prosjektplan med tidsestimat for hvert element som kan jobbes mot underveis.

Under utviklingen opprettes og testes hver modul for seg selv før den eventuelt blir koblet opp mot andre moduler som trenger data fra modulen.

1.5.1 Applikasjonsutvikling

Under arbeidet med applikasjonsutvikling ble verktøyene og plattformene Visual Studio Code, Node.js, localhost, OBS studio, SourceTree og GitHub benyttet.

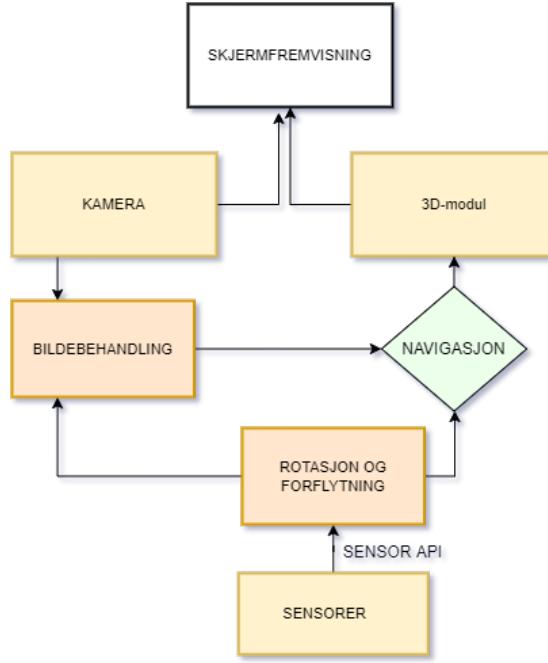
Selve kodingen foregikk i Visual Studio Code sitt redigeringsprogram[24]. Node.js[25] ble brukt for å kunne kjøre koden og localhost ble brukt til å kjøre koden lokalt på datamaskinen under utvikling av programvaren. Det ble brukt både webkamera og OBS sitt virtuelle kamera under lokal testing og utvikling av koden. Fordelen med å bruke OBS virtual kamera er at den kan erstatte webkamera og blant annet hente inn kamera fra smarttelefonen direkte inn i nettleseren lokalt. Se figur 2.



Figur 2: Applikasjonsutviklingen foregikk lokalt før den publisert på GitHub

SourceTree[26] er en Git-client som kommuniserer med GitHub og gjør det visuelt enklere å holde styr på hva som skjer i et git-repo inkludert versjonskontroll. Ferdig kode lagres i GitHub og kan hentes direkte til webapplikasjoner.

For å teste bruk av sensorene på smarttelefonen ble det nødvendig å publisere applikasjonen slik at den ble tilgjengelig ved hjelp av smarttelefonens nettleser. Heroku[27] har en innstilling hvor den henter koden direkte fra et valgt git-repo. Dette gjør det enkelt å publisere den mest oppdaterte koden fra det lokale utviklingsmiljøet gjennom SourceTree og GitHub.



Figur 3: Applikasjonsstrukturen som viser hvor de ulike modulene sender data.

1.5.2 Applikasjonsstruktur

For å kunne utvikle moduler knyttet til en infrastruktur, er det nødvendig å ha oversikt over alle modulene. Se Figur 3. Hvilke moduler skal inkluderes, hva skal de inneholde av funksjoner, hvilken data den trenger og hvor den skal sendes. Applikasjonsstrukturen ble lagd etter kravene til implementering av virkeligheten ved hjelp av kamera, en 3D-modul og tilgjengelige sensorer. Kamera og 3D skal begge vises, men sensorene brukes til navigasjon i 3D-rommet og til bildebehandling.

1.6 Prosjektplan

Prosjektet foregår over en periode på ett semester og har et forventet arbeidsomfang på omtrent 540 timer. Prosjektet ble påbegynt våren 2021, men grunnet personlig helse ble prosjektet satt på pause fra mars og innleveringen utsatt til desember 2021. Fristen for å levere forprosjektet var satt til slutten av februar, mens innlevering av selve prosjektet i begynnelsen av juni. Arbeidet i januar og februar bar preg av mye research til forprosjektet og utforming av mål for prosjektet.

I vårsemesteret ble det laget test-applikasjoner med AR-programvare i Unity for å bli kjent med applikasjoner med den aktuelle teknologien. Grunnleggende oppbygning og grunnleggende design ble skissert eller testet i separate script. Oversikt over planlagt arbeidsperiode og delmål i samme periode finnes som et Ganttdiagram i vedlegg A - "Prosjektplan Våren 2021". Arbeid med selve endeproduktet var såvidt påbegynt før august. Etter en relativt lang pause i arbeidet fra mars til august var det nødvendig å ha oversikt og forkaste utkast som ikke var egnet til videreføring i prosjektet. I august ble prosjektfilene flyttet til et nytt git-repo. Kode som var egnet til å jobbes videre med fra den første prosjektperioden i januar og februar ble overført til det nye repoet.

I løpet av høsten ble det gjort flere erfaringer, prosjektets mål bedre definert og derfor ble også prosjektplanen revidert med nye delmål. Den reviderte planen for høsten ligger i vedlegg B - "Prosjektplan Høsten 2021". Det ble lagt inn enkle beskrivelser av deloppgavene. Underveis i ukene i høstperioden ble det jobbet parallelt med koding på programvaren samtidig som det ble nødvendig å sette seg inn i aktuelle biblioteker. De første ukene i høstperioden gikk etter planen, men i perioder krevde deloppgavene mer testing og research som gjorde at resten av oppgavene ble forskøvet litt utover. Blant annet arbeidet med akselerometeret og OpenCv var større enn antatt. At det var planlagt mye tid til rapportskriving gjorde at det gikk fint å fordele og flytte på deloppgaver underveis.

Prosjektets krav ble satt i samråd med veiledere Morgan Konnestad og Jostein Nordengen, Universitetet i Agder gjennom veiledningsmøter i vårsemesteret. Prosjektet er i utgangspunktet tenkt som et gruppeprosjekt, men grunnet personlig helsesituasjon og valg av tema ble prosjektet løst i sin helhet som en gruppe på én.

1.7 Møtereferater

Møtereferater fra veiledningsmøtene ligger i Vedlegg C. Møtereferatet fra 9.mars 2021 konkluderer med at Forprosjektrapporten(vedlegg E) var godkjent og at alt var klart for videre arbeid etter prosjektplanen. All kommunikasjon etter dette tidspunktet foregikk på e-post og det eksisterer derfor ikke møtereferat som viser til avgjørelsen om å utsette innleveringen av prosjektet til desember 2021. Det ble ikke gjennomført veiledningsmøter høsten 2021.

1.8 Timeliste

Timelisten ligger i vedlegg D. Timeføringen er et omtrentlig antall timer jobbet med prosjektet i perioden. Enkelte perioder ble det ikke registrert arbeidstimer da det ble glemt. Det ble da ført opp et omtrentlig antall ut i fra hvor mye som ble utført på prosjektet i den perioden eller ut ifra tidspunktene det ble det lagt til endringer i prosjektkoden.

Det forventede arbeidsomfanget er på 540 timer over prosjektperioden. Det totale timer ført inn i timelisten i løpet av prosjektperioden er 560 timer.

1.9 Rapportstruktur

Kapittel 2 - *Teori og metoder* går i gjennom nødvendige teori og forkunnskaper for å kunne gjennomføre prosjektet.

Kapittel 3 - *Løsningsmetode* belyser enkelte utvalgte metoder tatt i bruk i prosjektet. Hvilke delmål og krav som er lagt til grunn for valget av metode blyses før det går inn på hvordan metoden benyttes konkret i prosjektet.

Kapittel 4 - *Resultat* går kortfattet gjennom resultatet av de valgte metodene. Applikasjonens struktur, funksjonalitet og innhold blir belyst og illustrert.

Kapittel 5 - *Diskusjon* drøfter resultatene opp mot prosjektets problemstilling og mål. Det vil også gjøres en vurdering av i hvilken grad metodene som ble brukt var egnet til å benyttes i en applikasjon med Augmented Reality-funksjonalitet.

Konklusjonen av diskusjonen og prosjektets gjennomføring ligger i kapittel 6 - *Konklusjon*.

2 Teori og metoder

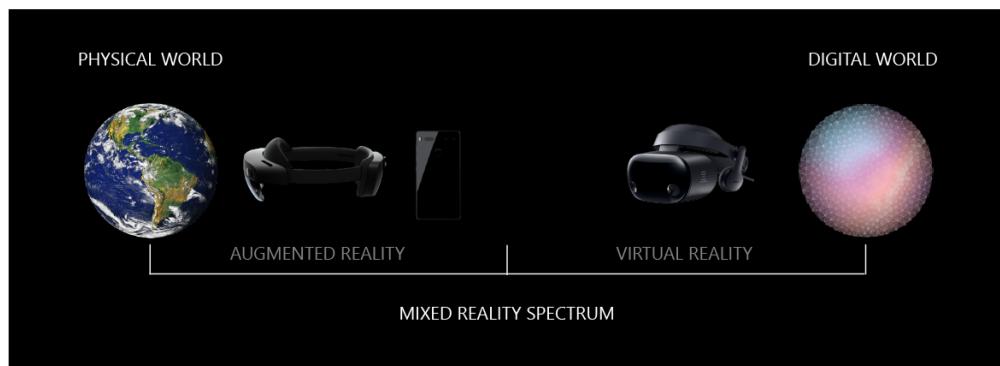
I dette kapittelet beskrives teorien bak de ulike systemene, metodene og teknologien som har betydning for videre gjennomføring av oppgaven. Det beskrives først hvor Augmented Reality blir plassert innenfor teknologien Extended Reality og hvilke komponenter som er essensielle for Augmented Reality. Videre forklares Augmented Reality teknologien detaljert hvor elementer som hardware, navigasjon og bevegelse, 3D, bildebehandling og tracking er sentralt. De ulike løsningene som kan benyttes under applikasjonsutvikling er også tatt med som en liten orientering for valgene som blir tatt senere.

2.1 Extended Reality

Extended Reality (XR) er et samlebegrep for teknologi som knytter sammen virtuell og fysisk realitet. Forskjellig sensor teknologi danner grunnlag for XR og benytter seg også av data teknologi eller annen teknologi som for eksempel briller [28]. Nancy Gupton deler teknologien inn i undergruppene Augmented Reality(AR), Virtual Reality(VR) og Mixed Reality(MR)[29].

Virtual Reality sender brukeren inn i en virtuell verden og stenger virkeligheten ute. Det benyttes briller og headsett for å eksponere brukeren for den virtuelle verdenen gjennom flere av sansene som hørsel og syn.

Augmented Reality kombinerer den digitale og virkelige verdenen gjennom en enhet med kamera, skjerm og nødvendige sensorer. Teknologien er mye benyttet blant annet som filter i Snapchat eller grafisk i Pokemon GO[30] hvor grafikk legges over skjermen og festes til et punkt eller objekt som er synlig gjennom kameraet.



Figur 4: Mixed Reality definert som spekteret av teknologi mellom det digitale og virkeligheten. Illustrasjonen er hentet fra Microsoft[3]

Mixed Reality har, som navnet tilsier, mål om å blande sammen virkeligheten og det virtuelle. Av The Franklin Institute blir MR definert som en kombinasjon av AR og VR[29], mens fire Microsoft utviklere bruker uttrykket Mixed Reality om spekteret mellom det virtuelle og virkeligheten[3]. De henviser til holografiske enheter - det vil si briller med gjennomsiktig skjerm som kan vise grafikk, dersom man ønsker å legge til digitale elementer

i virkeligheten. VR-enheter vil være i motsatt ende av spekteret og ta brukeren helt ut av virkeligheten ved at brillene blokkerer den ute.

2.2 Augmented Reality

Augmented Reality er for de fleste kjent som en teknologi som kan vise virtuelle objekter i den virkelige verdenen gjennom en skjerm, men hva ligger bak teknologien? På fagspråket sies det at AR har 3 viktige karakteristiske trekk [31]:

- AR kombinerer virtuell og virkelig data
- AR er interaktiv i sanntid
- AR operer og brukes i et 3D-miljø

Google introduserer Augmented Reality(AR) [32] som en teknologi som legger digitalt innhold og informasjon over den fysiske verdenen ved hjelp av sensorer. I Februar 2015 utførte Dhiraj Amin og Sharvari Govilkar et “sammenlignings studie av utviklingssystemer(SDK) for AR”[33]. De deler AR systemer inn i tre steg bestående av analyse, sporing og miksing. Ved å analysere bilder fra kameraet gis det mulighet til på gjenkjenne data fra virkeligheten. Metoden benytter seg av gjenkjennelige elementer som kan spores og det gir dermed mulighet for å plassere digitale elementer som grafikk med tilknytning til virkeligheten. Analysen gjøres gjennom markører, sensorer eller ved å gjenkjenne karakteristiske trekk i det fysiske miljøet.

Apple har også produsert et rammeverk for utvikling av AR - ARKit[16]. De viser til allerede eksisterende applikasjoner produsert med ARKit som er tilgjengelig for sine brukere. Fysioterapi, live bildemanipulering, lær om Apollo-romferder, gå mellom korallrev, spille kjente dataspill på ditt eget stuebord, diagnostisere plantene i hjemmet eller innrede huset med nye møbler før du bestiller dem. I tillegg bruker de mer kjente applikasjonen Snapchat og Instagram også AR teknologi til å manipulere omgivelsene eller ansiktene ved hjelp av filtre.

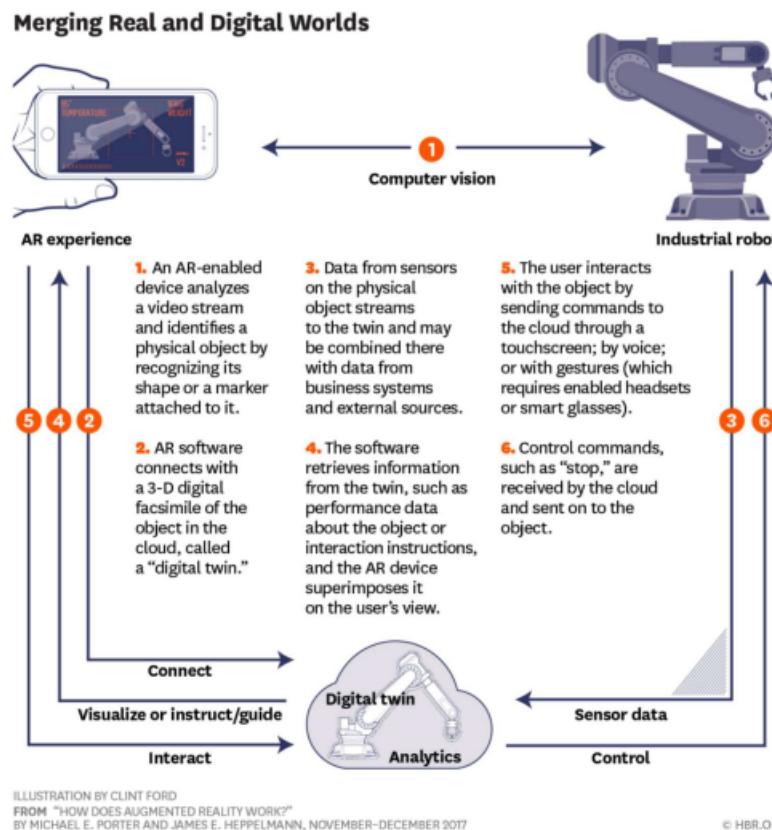
AR systemer kan ta i bruk markører eller sensorer for å beregne objektenes relative posisjon prosjektert på enhetene. Zeal AR er et i følge dem selv “en ledende AR utvikling service leverandør”[34] De ramser opp følgende komponenter som essensielle for AR systemene sine:[35].

- Hardware: Prosessorer, sensorer, input enheter, en skjerm for fremvisning.
- Software: Programvare som behandler inndata og muliggjør prosjektering av data og objekter.
- Remote server: Serverene benyttes for å rendre og lagre dataen som kan endres og hentes av applikasjonene.

Artikkelen “How Does Augmented Reality Work?”[4] ble skrevet for The Magazine i slutten av 2017, og publisert på Harvard Business Review sine sider om teknologi og analyse. Artikkelen er skrevet av en professor på Harvard Business School - Micheal

E. Porter, og James E. Heppelmann[36] - president og CEO av Parametric Technology Corporation(PTC)[37]. PTC er produsent av blant annet Vuforia, som en av de største AR plattformene og utviklingsverktøyene innen AR.[38] Forfatterne forklarer stegvis hvordan komponentene henger sammen og kommuniserer for å skape en realitetsnær opplevelse av de digitale objektene som blir lagt over bildene.

Michael E. Porter og James E. Heppelmann benytter seg av de samme komponentene som Zeal AR[35], men drar fram at opplevelsen av de implementerte 3D-modellene er grunnleggende for at det skal oppleves virkelighetsnært. Interaktive funksjoner og kontroller som sender data gjennom softwaren til serveren. Serveren kan oppdatere modellene etter mottatt data ved hjelp av sensorer eller kontroller, dette gir også en ekstra dimensjon til de digitale modellene. Se figur 5.



Figur 5: Illustrasjon av Clint Ford for artikkelen “How Does Augmented Reality Work?”[4]

Fra artiklene referert til over så forstår en at AR er en teknologi som omfatter mye og består av flere viktige metoder. Alle disse metodene blir videre i kapittelet forklart for å danne en forståelse av de viktigste punktene. Ser en på AR i det store bildet så avhenger alt av hvilken hardware som blir tatt i bruk. Det er helt avgjørende at riktig hardware er på plass, dette blir forklart i kapittel 2.3. Det neste som er viktig for AR omhandler applikasjonsdesign og er stegvis beskrevet i kapittel 2.4, 2.5 og 2.6.

2.3 Hardware

En optimal AR applikasjon vil inkludere hardware med god prosessorkraft, høyoppløselig kamera og skjerm, samt bevegelsessensorer med god nøyaktighet og hurtighet.

De aller vanligste bruksenheterne som kan kjøre en AR applikasjon i dag er smarttelefoner, nettbrett eller smartbriller. Samtlige av disse enhetene har som regel innebygd mye av hardware en trenger for å ta i bruk AR. Selv om dette kommer som en standard så vil ikke det si at alle enheter vil virke like bra. Faktorer som fører til at en smarttelefon er bedre enn en annen til å kjøre en AR applikasjon er viktig å forstå og for å kunne designe AR applikasjoner.

Dårlig oppløsning på kamera vil direkte påvirke hvor nøyaktig AR applikasjonen klarer å gjenkjenne elementer i bildet som igjen påvirker nøyaktigheten til ankerpunkt mellom den reelle og virtuelle verden. Dette medfører at 3D objekter plasseres feil eller vil ha feil skalering i forhold til den reelle verden. En liten skjerm eller en skjerm med dårlig oppløsning vil mest påvirke brukeren sin opplevelse av applikasjonen. Det kan bidra til å gjøre det vanskelig for brukeren å se de virtuelle objektene nøyaktig.

Det som vil påvirke en AR applikasjonen aller mest er prosessorkraften til enheten. Tunge beregningene, bildebehandling og 3D rendering krever mye av en prosessor og vil være avgjørende for helhetsinntrykket til AR applikasjonen. En dårlig prosessor vil medføre at oppdateringshastigheten på skjernmbildet oppleves hakkete og vil direkte påvirke nøyaktigheten til hele applikasjonen. Se kapittel 2.5 og 2.6 for mer detaljer om henholdsvis brukernavigasjon og 3D rendering.

2.4 Applikasjonsutvikling

2.4.1 Mobilapplikasjon

En mobilapplikasjon er programvare som lastes ned på smarttelefonen, nettbrett eller andre mobile enheter. Mobilapplikasjonen har den fordelen at brukergrensesnittet må oppnå en standard både i utvikling og kvalitet før den kan bli publisert og kan kjøres på operativsystemene. Ulempen er at applikasjonen kun kan kjøres på enheten som faktisk har lastet ned programvaren og har et operativsystem som møter de samme kravene som applikasjonen er bygget for. Det betyr at en applikasjon bygget for iOS kan derfor ikke kjøre på en Android-enhet eller motsatt. I tillegg kan applikasjonene opppta vesentlige andel av lagringsminnet til enheten.

2.4.2 Web applikasjon

Webapplikasjoner er bygget med HTML og CSS og krever mye mindre av enheten. Applikasjonen kjøres gjennom en nettleser og informasjon og data lagres på en server-basert database. Applikasjonen blir ikke lastet ned på enheten men kjøres ved hjelp av en skybasert plattform rett i nettleseren. Webapplikasjoner kan benytte seg av Web APIer som gir tilgang[23] funksjoner som bidrar til infrastrukturen. Javascript er et av de vanligste programmeringsspråkene brukt i webapplikasjoner.

2.4.3 Skytjenester

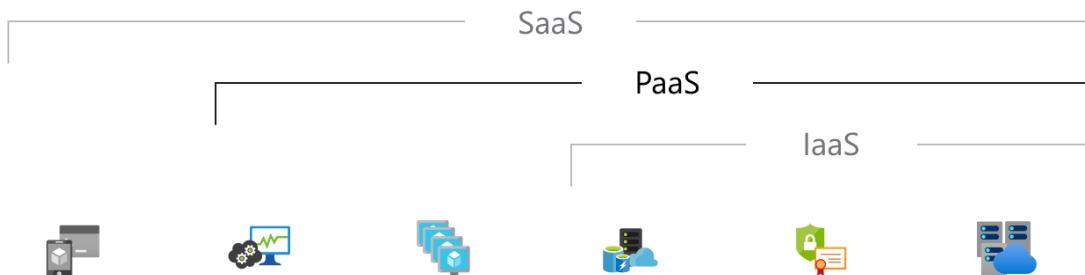
En applikasjon trenger en infrastruktur med databaser og servere for å kunne kjøre. Applikasjoner som kjører via servere i skytjenester blir tilgjengelige for forbrukerne og fjerner behovet for å installere lokale servere, databaseprogramvare og nødvendig programvare for operativsystemet lokalt på enheten. Leverandører av skytjenester som Microsoft deler tjenestene i ulike nivåer tilpasset forbrukere og utviklere etter behov.

Dersom leverandøren eier hele applikasjonen som man kobler seg opp mot, for eksempel en mail, kalender eller et skriveprogram gjennom Microsoft 365 - benytter man seg av **“Software as a service”** (SaaS)[39]. Brukeren betaler for å kunne bruke applikasjonen som kan settes opp raskt. Alt av vedlikehold, lagring og drift drives av tjenesteleverandøren som kan sikre at applikasjonen er tilgjengelig og sikker i bruk.

En **“Platform as a service”** (PaaS)[40] løsning gir brukeren mulighet til å utvikle og vedlikeholde applikasjoner selv, mens leverandøren av plattformen står for infrastrukturen som inkluderer database, operativsystemer, servere og lagring. Dette gir mulighet for å utvikle enkle applikasjoner uten å måtte tenke på om systemet bak fungerer.

“Infrastructure as a service” (IaaS)[41] er ideelt for en bedrift som vil ha fleksibilitet under utvikling og drift vedlikehold av applikasjonene selv. Det er valgfrihet på hvilke deler av infrastrukturen de ønsker å sette bort til leverandøren av tjenesten, og det mulig å redusere kostnader på nødvendige tjenester som servere og brannmurer.

“Serverless computing”[42] er en tjeneste som har som mål å hjelpe utviklere med å bygge applikasjoner og infrastruktur raskere og bedre ved å tilby tilleggtjenester på alle nivåer i infrastrukturen.



Figur 6: Modell laget av Microsoft som beskriver hva skytjenestene vanligvis inneholder

Heroku[27] er en online plattform som brukes for å kjøre applikasjoner på web. De tilbyr Paas løsningen som gir fleksibilitet til å bygge og utvikle applikasjoner uten å tenke på infrastrukturen bak. Heroku leser gjennom kildekoden, konfigurasjonsfiler, bygger opp og kjører applikasjonene på sin plattform. Plattformen har både dokumentasjon for installasjon og støtter Node.js[25], som kjører Javascript på servere. En applikasjon kan kobles direkte på et GitHub-bibliotek for å enklere kunne oppdateres med de nyeste endringene under utvikling. I tillegg er det mulig å utvide med tilleggstjenester som gir tilgang til blant annet databasen.

2.5 Navigasjon og bevegelse

Navigasjon og bevegelse gjør det mulig for applikasjoner å være interaktive. Å kunne aktivere funksjoner eller animasjon gjennom å hente informasjon om brukerens bevegelse og interaksjon med enheten gjør det mulig å gi brukeren en mer levende og interaktiv opplevelse.

2.5.1 Knapper

En vanlig navigasjonsmetode i applikasjoner er ved tastetrykk eller mus. Disse trykkene kan aksesseres gjennom et “mouse”- eller “keydown”-event. Hver tast har en egen lokasjon og kode som kan brukes til å identifisere hvilken tast som er trykket på. Ettersom dagens smarttelefoner og nettbrett benytter seg hovedsakelig av berøring direkte på skjermen gjelder dette hovedsakelig applikasjoner til bruk på datamaskiner.

2.5.2 Berøring

For å hente informasjon fra berøringssensitive skjermer (“touch”-skjerm) kan man benytte et “touch”-event. Dette eventet utgir en posisjon fra DOM-elementet, og kan fortelle om når berøringen startet, om den pågår og registrerer at det avsluttes.

2.5.3 Sensorer

Fremskritt innen nanoteknologien med MEMS eller “mikro-elektrø-mekaniske systemer”[17] har gjort det mulig å få mer og mer avanserte sensorer inn i hverdagens teknologiske enheter, blant annet smarttelefoner. Størrelsen til MEMS kan variere fra en mikrometer til noen milimeter[17]. I følge Finn Jarle Kvalheim som har skrevet artikler for tek.no[43] inneholder “alle” smarttelefoner følgende sensorer: Elektronisk kompass, magnetometer, akselerometer, gyroskop, GPS-mottaker, CMOS-sensor og kapasitiv fingersensor. Andy Cho fra Samsung SDS bekrefter i artikkelen “What Kinds of Sensors are Embedded in Smartphones?”[44] fra 2020 at smarttelefonene inneholder sensorer som registrerer både bevegelse, gravitasjon, enhetens orientering, lokasjon og data om omgivelsene som lys, trykk, temperatur og fuktighet.

Bevegelsessensorene er de som er mest sentrale med tanke på bevegelse, men også lokasjonsendringer og orientasjon kan benyttes til navigasjon i applikasjoner som for eksempel karttjenester eller det kjente spillet Pokémon Go[30] hvor brukeren beveger seg over store geografiske områder.

Sensordataen kan hentes inn i koden ved hjelp av APIer eller objektmetoder.

Akselerometeret er en bevegelsessensor som kontinuerlig registrerer akselerasjonen langs sensoraksene og kan fange opp bevegelse og vibrasjoner [45]. Verdiene mottas i positive og negative x-, y- og z-verdier i antall meter per sekund i andre(m/s^2). Måten et MEMS akselerometer klarer å hente ut akselerasjonen på er ved å ta i bruk Newton's 2nd lov, $F = m * a$ hvor F = kraft, m = masse og a = akselerasjon. Med en kjent masse i akselerometeret kan dermed akselerasjonen beregnes ved: $a = F/m$.

Akselerasjon er det samme som endring i hastighet per tidsenhet. Hastighet er igjen avstanden noe har forflyttet seg per tidsenhet. Posisjonsendring kan derfor i prinsippet beregnes matematisk ved å ta dobbelt integralet av akselerasjonsmålingen. Integral er det samme som å summere opp funksjonsverdier i et gitt intervall, og metoden kan dermed først benyttes til å summere alle små endringer i akselerasjonen per tidsendring for å få hastighet (se ligning 1). Endring av hastigheten kan da igjen summeres per tidsendring for å få posisjon (se ligning 2). Sammen med akselerasjonsmålingene vil det alltid komme en konstant missvisning (bias). Dobbeltintegral av akselerasjonsmåling vil derfor introdusere en missvisning som vokser eksponentielt over tid. Denne metoden kan derfor i mange tilfeller avvike mye fra den reelle posisjonen.

$$v = \int a \cdot dt \quad (1)$$

$$s = \int v \cdot dt \quad (2)$$

hvor:

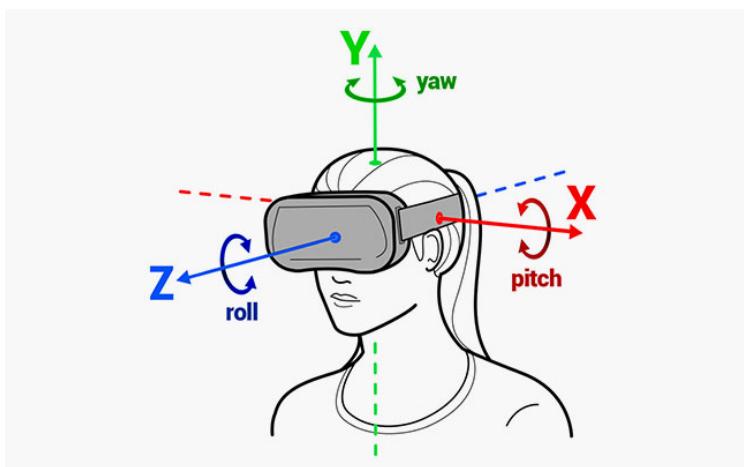
$$\begin{aligned} v &= \text{hastighet } [m/s] \\ a &= \text{akselerasjon } [m/s^2] \\ dt &= \text{tidsendring } [s] \end{aligned}$$

hvor:

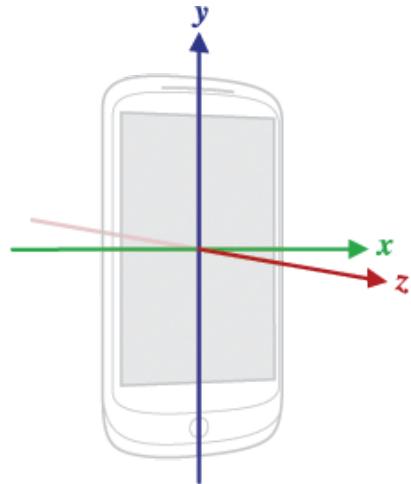
$$\begin{aligned} s &= \text{posisjon } [m] \\ v &= \text{hastighet } [m/s] \\ dt &= \text{tidsendring } [s] \end{aligned}$$

Gyroskopet Gyroskopet er en bevegelsessensor som benytter seg av jordens tyngekraft for å kontinuerlig måle graden av rotasjon enheten har om sine egne akser.[46] Dataen sendes som verdier i x, y og z-aksene i radianer per sekund($\frac{rad}{s}$). Gyroskopet brukes i spillutvikling til styring av biler og fly, til å navigere ved å riste på smarttelefonen eller til å stabiliserer kamera ved å utligne bevegelser som risting.[47]

Magnetometeret er også en av de vanligste sensorene i mobile enheter. Sensoren kan måle enhetens orientering ved hjelp av jordens magnetfelt. Magnetometeret gir kontinuerlig data om orienteringen til enheten om de tre sensor-aksene [48]. Gjennom Window-objektet kan Event-handleren “deviceorientation” hente ut positive og negative verdier i x, y og z i antall grader enheten er rotert rundt aksene. Sensoren brukes i digitale kompass og for å lese orientasjonen i blant annet fly.



Figur 7: Bevegelsessensorer brukt til orientering i VR-teknologi[5].



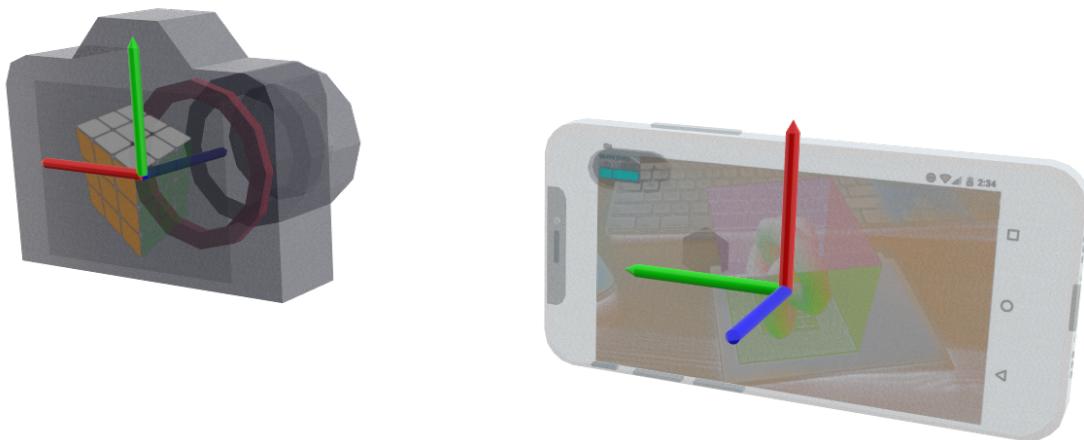
Figur 8: Smarttelefonens koordinatsystem hentet fra Android Developers[6].

2.5.4 Koordinatssystemer og akser

Mobile enheter benytter et lokalt koordinatsystem for å registrere retning på bevegelse og orientering[6]. Dette koordinatsystemet følger skjermens relative retning. Ettersom dette koordinatsystemet beveger seg med enheten kan i praksis være mye i forhold til visuelle eller reelle verdenskoordinater. Koordinatsystemene kan også være annerledes orientert enn andre elementer som innebygde sensorer. Dette er en faktor som må tas hensyn til når data fra sensorene benyttes i applikasjoner eller utregninger. Se figur 8.

2.5.5 Euler translasjon

I en applikasjon kan det finnes flere komponenter som har eget lokalt koordinatsystem. Transformasjon fra det ene koordinatsystemet til det andre kan gjøres ved hjelp av Euler translasjon [49]. I dette prosjektet benyttes det et kamera og sensorer fra en smarttelefon. Dataen sendes inn i et kamera-objekt i en Three-scene. Kameraet til Three-scenen har sitt lokale koordinatsystem og forskjellen kan sees i figur 9. Rød, grønn og blå pil representerer henholdsvis x, y og z aksen. Her kan en se at en rotasjon om x-aksen til mobilen vil tilsvare rotasjon om y-aksen til kamera.



Figur 9: Koordinatsystemet til kamera-objektet i Three-scenen og smarttelefonen.

Ved å benytte Euler translasjon kan en altså si at rotasjon rundt den ene aksen til et system skal tilsvare rotasjon om en annen akse i et annet system. Euler translasjonen fra mobilen til kamera-objektet blir derfor:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} Y \\ X \\ -Z \end{bmatrix}$$

2.6 3D-biblioteker for Javascript

På studiet Multimedieteknologi og -design er javascript valgt ut som script-språk. Javascript er et høynivåspråk og populært tatt i bruk på web.

Ved et raskt søk etter 3D rammeverk og biblioteker finner man at det finnes utallige små og store rammeverk og biblioteker for 3D i javascript. Går man ut ifra topp 5 på tilfeldige lister finnes det en allikevel en gjenganger. Selv om det ikke alltid kan garanteres at topplister som Slant[50], Dunebook[51] og 1stWebdesigner[52] vurderes og godkjennes av eksperter, så har det betydning for biblioteket at mange tar det i bruk. Et bibliotek som Three.js lastes ned 37000 ganger per uke i følge Openbase[53]. Utviklere verden over tar i bruk biblioteket og det havner samtidig så høyt blant vurderingene fra forbrukerene. Det er en god indikator på at det finnes eksempler, god dokumentasjon og at det er et godt gjennomført og vedlikeholdt bibliotek som fungerer.

	Dunebook	Slant	1st-Webdesigner	Openbase Anbefalt	Openbase Best rating	Openbase Most used (weekly DL)	Openbase Vedlikehold
1	Three.js	Three.js	Three.js	Three.js	gsap	Three.js(377K)	Three.js
2	D3	Babylon.js	Babylon.js	cesium	Three.js	gsap(45K)	cesium
3	Aframe	PlayCanvas	Cannon.js	xeokit-sdk	cannon	deck.gl(44K)	xeokit-sdk
4	Babylon.js	CopperLight	CopperLicht	deck.gl	xeokit-sdk	cesium(19K)	plurid-react
5	Zdog	JS3D	Phoria	gsap	react-babylonjs	react-three/(6K)	deck.gl

Figur 10: Three.js øverst på 6 av 7 vurderinger av rammeverk for 3D i Javascript

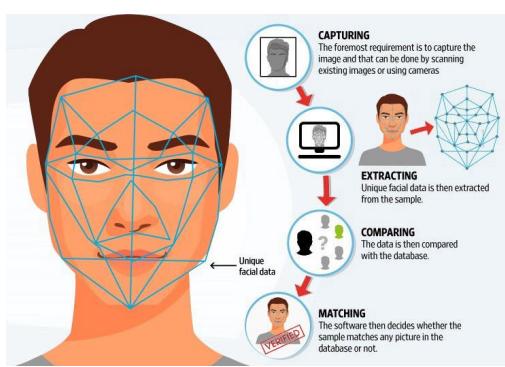
2.6.1 Three.js

Three.js er et javascript-bibliotek for 3D-implementering i nettlesere ved hjelp av WebGL-rendering[54]. WebGL er et grafisk bibliotek bygget for å få interaktiv 3D-grafikk inn i nettlesere med javascript. Nettsiden threejs.org[55] inneholder dokumentasjon, eksempler og et programmeringsgrensesnitt, mens kildekoden ligger tilgjengelig på nettsiden og GitHub. Modulen til Three.js inneholder muligheter for å opprette en scene med ulike kameratyper. Det er muligheter for å legge til lys, skjeletter, geometriske objekter og hjelpeobjekter i scenen.

2.7 Bildebehandling og tracking

Kameraet er en av de mobile enhetenes hovedsensorer som blir brukt i Augmented Reality. Sensoren gir tilgang til å finne markører eller gjenkjenne konturer av objekter i virkeligheten. For å analysere data gjennom kameraet brukes det ulike teknikker som enten ser etter konturer eller objekter for å analysere miljøet enheten befinner seg i. Dataen fra analysene i slike teknikker kan brukes som ankerpunkter for å plassere objekter over bildet og inn i miljøet.

Digit.in[8] starte med å nevne **gjenkjenning basert AR** som en av de vanligste typene av AR. Programvaren gjenkjenner objekter som bilder, objekter, ansikter eller markører. Under prosessen hentes deretter informasjon eller data om objekter som deretter sendes til skjermen. Markører brukes mest som et ankerpunkt for modeller som plasseres ut over markøren. Markører kan erstattes med for eksempel levende modeller som i figur 12. Ansiktsgjenkjenning brukes både som sikkerhet på enheter og for manipulering gjennom filtre.



Figur 11: Ansikt gjenkjenning. Illustrasjon fra Huawei Developers for Medium.com[7]



Figur 12: QR-kode markør som erstattes med en levende modell. Bilde:digit.in[8]

Uten kjente objekter tilgjengelig kan det allikevel være mulig innen AR å lage sine egen ankerpunkter. Det finnes teknikker for å hente ut både plan, kanter, siluetter, hjørner og linjer fra et bilde. Disse trekkene kan igjen brukes som ankerpunkter for modeller, tekst eller annen grafikk. En av de mest kjente teknikkene er Edge Detection.

2.7.1 Edge Detection

Edge Detection, eller kant-deteksjon, er en metode som brukes for å kunne analysere kantene til objekter i et bilde. Som vi vet består bilder av en stor matrise som inneholder informasjon posisjon og lysstyrke for hver piksel. Disse egenskapene gjør at pikslene kan sees i sammenheng med hverandre og danne konturer som er gjenkjennelige for metoden. I følge OpenCV.org[22] er kanter eller overganger mellom objekter en av de viktigste delene av et bilde. Kanter kan fortelle om størrelser eller form til objekter og derfor er analyse av kanter “Edge detection” mye brukt innen bildebehandling.

For å finne en kant i et bilde må en se etter en rask endring av lysstyrke i pikslene. Det er nettopp denne metoden som brukes i Edge Detection og består av følgende steg [22]:

- Last inn bilde og gjør det om til et gråskala bilde
- Gjør bildet uskarpt ved hjelp av *GaussianBlur*
- Bruk *Sobel Operator* til å fremheve kanter og sette alle andre piksler som ikke er en del av en kant lik null

Den største variabelen til Edge Detection er lysforholdene. Ved store endringer i lysforhold vil det føre til at Edge Detection ikke klarer å skille ut kantene slik det er tenkt. Det er derfor nødvendig med gode lysforhold for at denne behandlingen skal gi utslag som kan benyttes til videre behandling og eventuelt som en markør.



Figur 13: Eksempel på bilde som en tar Edge Detection på

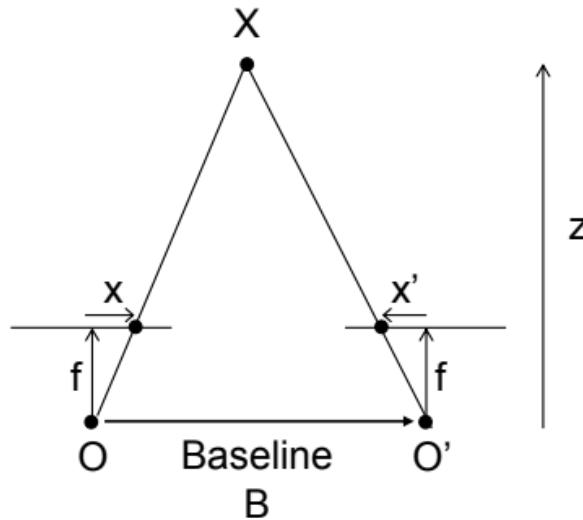


Figur 14: Edge Detection resultat

2.7.2 Dybdeberegning

Modulen i Augmented Reality som sørger for at den virtuelle verden har riktig størrelse i forhold til den reelle verden blir styrt av en *skaleringsfaktor*. ARCore refererer til et *Depth API* som brukes i deres kode [15]. Dette API-et danner et dybde-bilde som gjør det mulig å danne et forhold mellom objekter kamera ser og hvordan virtuelle objekter da skal forholde seg i forhold til det. ARCore påpeker også at *Depth API* krever hardware med god prosessor og nødvendige sensorer i enheten, noe som ikke alle dagens enheter har. Dybdesensorer er per i dag mer og mer tilgjengelig, men ikke vanlig i alle smarttelefoner helt enda.[43]

En annen metode for å finne *skaleringsfaktor* er ved bruk av stereosyn. Beregning gjøres ved hjelp av matematisk geometri med data fra to bilder og kameraets egenskap. I Figur 15 ser en parameterene som er nødvendig i beregningene. O og O' tilsvarer kamera 1 og 2, de to horisontale linjene tilsvarer bildeplanet. Punkt X er punktet en skal måle avstanden Z til, og B er avstanden mellom kamera 1 og 2.



Figur 15: Figur som beskriver dybdemåling ved stereosyn [9]

På fagspråket kalles metoden disparitet og refereres til som den inverse proporsjonale til dybden [9]. Videre ser formelen slik ut:

$$d \cdot s_c = x - x' = \frac{B \cdot f}{Z} \quad \text{Hvor:}$$

$$Z = \frac{B \cdot f}{d \cdot s_c}$$

d = disparitet

f = fokallengde [mm]

Z = Avstand fra kamera til objekt [mm]

B = Avstand mellom kamera 1 og 2 [mm]

x = pikselavstand fra bildeplanets sentrum og ut til hvor
objektet prosjekteres på i bildeplanet [pix]

s_c = Kamerasensor størrelseforhold [mm/pix]

3 Løsningsmetode

Som nevnt i kapittel 1.2 er målet å teste mulighetene for å oppnå grunnleggende AR-funksjonalitet i en applikasjon ved enkle og grunnleggende funksjoner i javascript og ved bruk av openSource-biblioteker. Løsningsmetodene tar for seg hvilke hvordan de valgte metodene benyttes og hvordan metoden bidrar til at resultatet oppnår kravene til applikasjonen.

3.1 Krav til løsningen

Overordnede krav er listet opp under 1.2 Problemdefinisjon. Videre følger punkter som oppsummerer hvilke funksjoner det er ønske om å teste og prøve å implementere. Disse er utarbeidet fra kravene til en Aumented Reality-applikasjon og tilgjengelige sensorer i tilgjengelige smarttelefoner.

- Inneholde en 3D-modul med mulighet for å implementere 3D-objekter.
- Skjermen skal inneholde fremvisning av bilder fra omgivelsene i sanntid.
- Interaktivitet skal baseres på data hentet fra sensorer og/eller kamera.
- 3D-miljøet skal motta data fra andre moduler eller script som påvirker objekter i miljøet.
- Applikasjonen skal kjøres som en webapplikasjon.
- Appplikasjonen skal benytte seg utelukkende av OpenSource-biblioteker.

Hvilke løsningmetoder skal benyttes for å nå delmålene?

- For å implementere et 3D-miljø i applikasjonen benyttes Three.js sitt bibliotek og modul.
- Ved hjelp av css er det mulig å tvinge flere HTML-elementer til å ligge over hverandre. Three.js skal ha en gjennomsiktig bakgrunn for at det skal være mulig å se både video og Three-objekter samtidig.
- Data fra magnetometeret og akselerometeret skal brukes til å påvirke objekter i 3D-scenen som oppdateres i sanntid.
- Objekter skalieres eller festes til markører synlige i kameraet gjennom bildebehandling og dybdeberegning.
- Det bygges en applikasjon på Heroku som kan brukes til å teste funksjonaliteten. Applikasjonen kjøres som en webapplikasjon.

3.2 Webapplikasjon

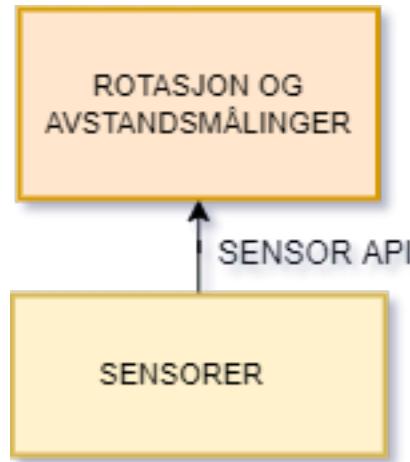
Det ble valgt å lage benytte løsninger for å lage en webapplikasjon for prosjektet. En webapplikasjon gir tilgang fra enheter uten at programvaren må lastes ned. Den trenger ikke ta hensyn til operativsystemene til enhetene da den kjøres rett i nettleseren.

3.2.1 Heroku

For å benytte seg av plattformen til Heroku opprettes det en konto. Hobby-applikasjoner og utprøving av konsepter kan opprettes gratis. Hver applikasjon kobles til et eget Git-repo som automatisk henter oppdatert kode fra repoet og kjører applikasjonen på en tilegnet nettadresse. Heroku-klient bruker terminalen for å lage en kobling mellom det lokale utviklingsmiljøet og plattformen ved hjelp av Herokus egne installasjonsinstrukser tilpasset bruk av Node.js [56].

3.3 Navigasjon og bevegelse

En AR-applikasjon krever at det inkluderes interaktivitet som drives av data som kommer fra omgivelsene. Dette kan gjøres gjennom å implementere enhetens bevegelse i den virkelige verdenen som bevegelse i det virtuelle rommet. Se eget kapittel om det virtuelle 3D rommet i kapittel 3.4.1.



Figur 16: Sensor API innholder funksjoner som henter informasjon fra sensorene.

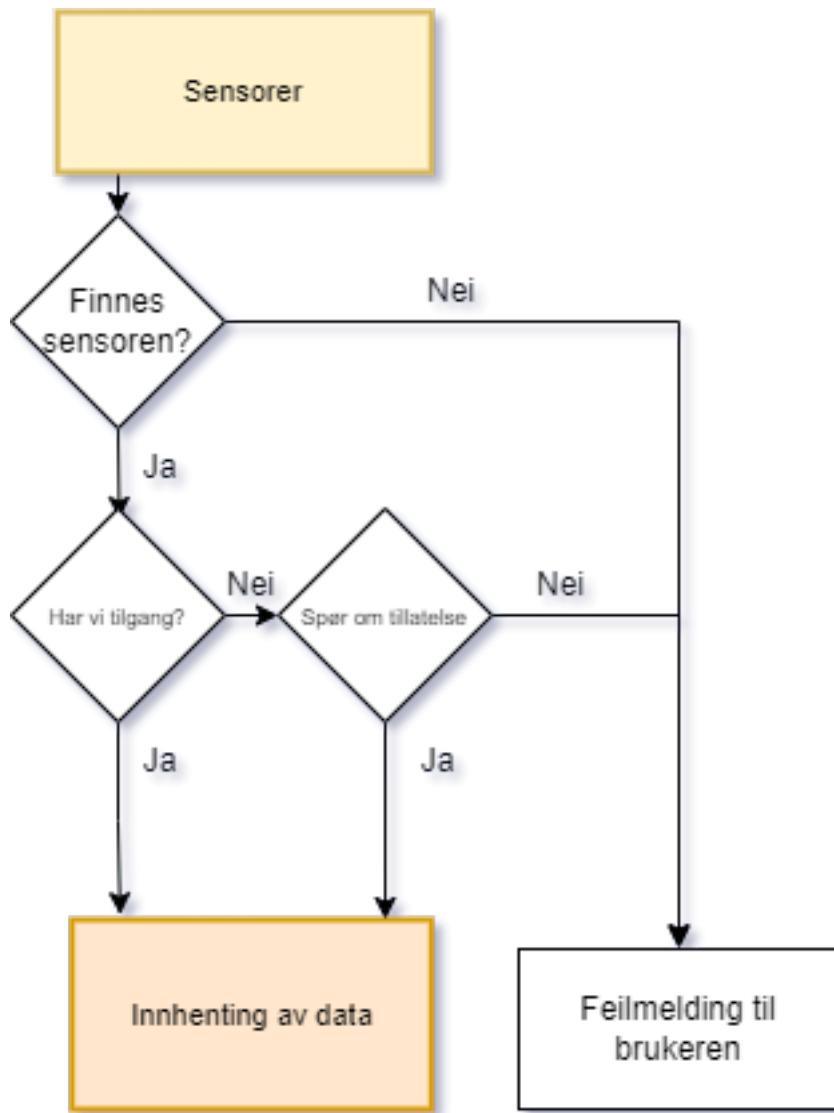
For å kunne hente data fra sensorene bruker vi et Sensor API[57]. Dataen hentes ved hjelp av en funksjon eller funksjoner som lytter etter triggere(“EventListener”).

```
window.addEventListener('deviceorientation', handleOrientation, true);
```

Figur 17: En “Event Listener” som lytter til endringen i orienteringen til enheten.

Sensorene er en del av hardwaren til enheten. Brukeren skal ha mulighet til å gi eller nekte applikasjoner tilgang til enhetens prosessorer, hardware og innhold. Det er derfor nødvendig å spørre brukeren om tillatelse til å hente data. Dette kan gjøres ved å benytte et permissions API[58], men ettersom dette API kun er kompatibelt med under halvparten av netteleserne vil manuell forespørsel være et alternativ.

Dersom sensorene eksisterer på enheten, sjekkes det om det er gitt tilgang. Dersom brukeren ikke har gitt tillatelse sendes det beskjed til brukeren om at applikasjonen mangler tillatelse til sensorene. Brukeren kan få en pop-up, eller bare bli referert videre til sitt innstettingspanel for nettsiden.



Figur 18: Flytskjema for å spørre om tilgang til sensorene ved bruk av permission API.

3.3.1 Magnetometer

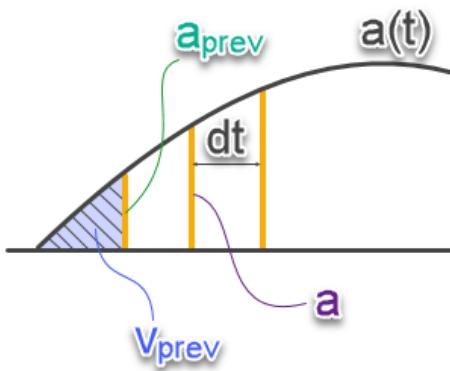
Magnetometeret er som beskrevet i kapittel 2.5.3 en orienteringssensor som er liten nok til å bli implementert som en del av teknologien i smartelefonene. Dataen fra magnetometeret er tilgjengelig gjennom egenskapene til “window”-objektet - et “deviceorientation” event. Rotasjonen om henholdsvis x-, y- og z-aksen til enheten aksesseres gjennom *event.beta*, *event.gamma* og *event.alpha*. For å få dataen i radianer må den regnes om fra grader ved å multiplisere med $\frac{\pi}{180}$.

For å ta dataen i bruk med en Three-modul eller andre script kan dataen lagres som en global variabel. Hvordan enhetens absolutte rotasjon kan benyttes i en Three-scene beskrives i kapittel 3.4.1

3.3.2 Akselerometer

På samme måte som magnetometeret kan dataen fra akselerometeret hentes ved hjelp av en EventListener - i dette tilfelle er det “devicemotion” som settes som event-handler. Akselerasjonen måles langs aksene x, y og z i m/s^2 . For å hente ut akselerasjonen inkludert gravitasjonen så kan man hente ut ved å bruke “event.accelerationIncludingGravity”. I dette prosjektet eksluderes gravitasjonen og derfor benyttes “event.acceleration”.

Fra matematikken vet vi at akselerasjon er definert som endring i hastighet per tidsenhet, og igjen at hastighet er endring av posisjon per tidsenhet. Vi vet dermed at ved å summere opp akselerasjon over et gitt tidsintervall vil gi ut hastighet. Samme gjelder posisjon som er summen av hastighet over et gitt tidsintervall.



Figur 19: Akselerasjonsmåling som en funksjon av tiden delt opp per tidsenhet (dt)

Ved å ta integralet av akselerasjon får vi altså hastigheten, og den dobbelt-integrerte av akselerasjonen er dermed lik posisjonen. Dette kan vi se ut ifra figur 19 og kan videre definere ligningene 3 og 4.

For å teste stabilitet og nøyaktighet av den kalkulerte posisjonen fra akselerasjonsdata, er det utført mange tester hvor forflyttet strekning har vært en kjent faktor. Data som er samlet fra testene er lagt inn i regneark og plottet for å se resultatet visuelt. Testene er beskrevet nærmere i kapittel 3.3.3.

$$v(t) = v_{prev} + a_{prev} * dt + \frac{a - a_{prev}}{2} * dt * dir_a \quad (3)$$

$$s(t) = s_{prev} + v_{prev} * dt + \frac{v - v_{prev}}{2} * dt * dir_v \quad (4)$$

Hvor:

a = akselerasjon [m/s^2]

a_{prev} = akselerasjonen fra forrige måling [m/s^2]

v = hastighet [m/s]

v_{prev} = hastigheten fra forrige måling [m/s]

s = posisjon [m]

s_{prev} = posisjonen fra forrige måling [m]

dir_a = akselerasjonens positive eller negative retning [+/-]

dir_v = hastighetens positive eller negative retning [+/-]

dt = tidsendring [s]

Akselerasjonen (a) er data som blir hentet inn fra akselerometeret. Den blir filterert før den benyttes i utregningene. Akselerasjonen blir lagret i en global variabel på samme måte som resultatet fra utregningen av hastigheten (v) og strekningen (s). De kan da brukes under neste utregning som a_{prev} , v_{prev} og v_{prev} . Tidendringen (dt) er en tidsvariabel som måles hver gang eventet kjøres. Retningen til akselerasjonen(dir_a) og hastigheten(dir_v) hentes ved å sjekke verdien på henholdsvis akselerasjonen og hastigheten. I formlene erstattes retningen med enten ± 1 eller 0

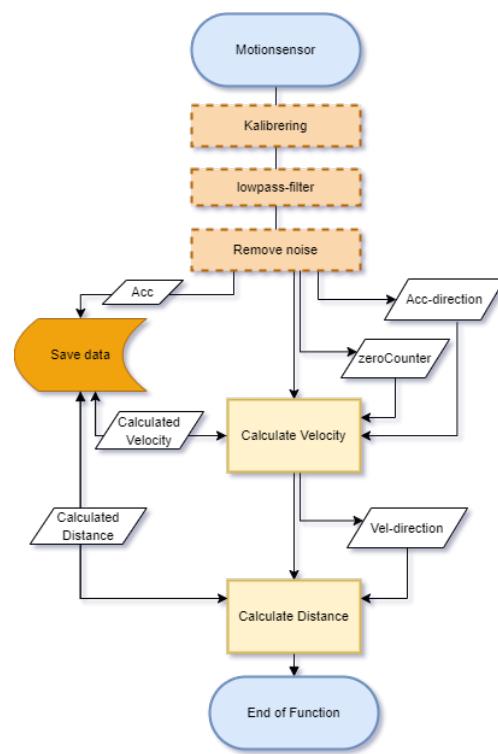
Grunnet mye støy eller feil på målingene vil det være nødvendig å filtrere dataen før den kan benyttes videre.

Kalibrering av dataen kan gjøres ved å innhente data som blir registrert når enheten ligger i ro. For så å trekke fra et en gjennomsnittsmåling i fremtidige målinger.

Lowpass-filter slipper igjennom signaler med lave frekvenser og filtrerer bort høyfrekvente signaler som er der den verste støyen. Uten tilgang til akselerasjonens frekvenser blir det i stedet laget en funksjon som jevner ut målingene ved at akselerasjonen blir satt til gjennomsnittet av de siste målingene.

I **Noise-filteret** settes det en grense for hvilke målinger som skal gjøre utslag. Målinger som er under satte grense blir satt til null og vil ikke påvirke målingen videre. Dette motvirker litt av driftsenhet som kommer i integreringsprosessene.

Akselerasjonen er null når det ikke er endring i hastigheten. Allikevel vet vi at null-akselerasjon ikke alltid samsvarer med null i hastigheten. Med store usikkerheter i akselerasjonsmålingene hender det at hastigheten ikke utlignes og man opplever drift over tid. Det er nesten umulig å holde jevn hastighet på en enhet over tid, og derfor kan vi si at vedvarende akselerasjonsmålinger lik null over tid gir hastighet lik null. For dette benyttes det en **ZeroCounter** som teller null-akselerasjonen. Når ZeroCounteren oppnår en satt verdi vil hastigheten også bli satt til null, så lenge akselerasjonen ikke er lik null vil ZeroCounteren nullstilles.



Figur 20: Oppbygning av handleMotion.js som henter data fra akselerometret og regner ut farten og strekningen.

3.3.3 Test av akselerometeret til avstandsberegning

Det ble utført tester av akselerasjonsdataen for å tilpasse filtere og utregningsmetode for å få data om tilbakelagt strekning. Enheten ble lagt i ro og applikasjonen startet på nytt før enheten ble flyttet 30cm langs en av aksene. Dataen ble lastet ned i CSV-filer for å visualisere og analysere utviklingen av dataen over tid. Testene ga tilgang til å lage grafer med og uten filtere, samt gjøre justeringer etter oppnådde resultater.

Testen ble utført ved å flytte smarttelefonen langs x-aksen over en oppmålt strekning på 30cm. Deretter ble dataen printet ut i en csv-fil for analyse. Formlene som ble brukt ble kontrollert ved å gjennomføre en utregning ved bruk av eksempladata.

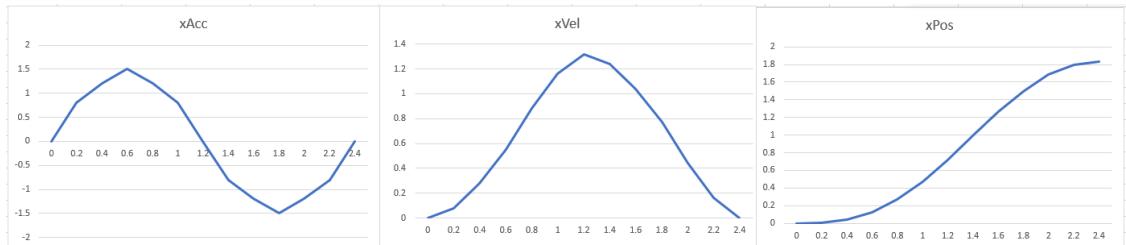
	A	B	C	D	E	F	G
1	Time	dt	acc	dir	xVel	Dir	xPos
2	0	0	0	0	0	0	0
3	0.2	0.2	0.8	1	0.08	1	0.008
4	0.4	0.2	1.2	1	0.28	1	0.044
5	0.6	0.2	1.5	1	0.55	1	0.127
6	0.8	0.2	1.2	1	0.88	1	0.27
7	1	0.2	0.8	1	1.16	1	0.474
8	1.2	0.2	0	0	1.32	1	0.722
9	1.4	0.2	-0.8	-1	1.24	1	0.994
10	1.6	0.2	-1.2	-1	1.04	1	1.262
11	1.8	0.2	-1.5	-1	0.77	1	1.497
12	2	0.2	-1.2	-1	0.44	1	1.694
13	2.2	0.2	-0.8	-1	0.16	1	1.8
14	2.4	0.2	0	0	0	0	1.832

Figur 21: Eksempladata for kontroll av formlene



Figur 22: Enheten ble flyttet 30cm langs x-aksen.

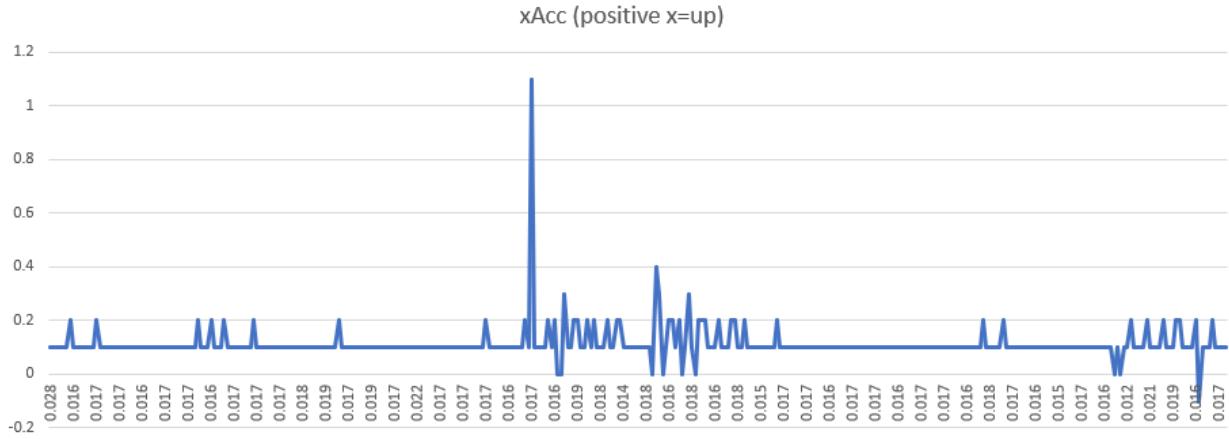
Grafene(figur 23) laget ut fra eksempladataen gir forventet resultat ut ifra at hastigheten er integralet av akselerasjonen og strekningen integralet av hastigheten. Ved så sjeldne målinger som i eksempelet ville feilkilden blitt veldig stor og målingene foregår derfor mye tettere enn i eksempelet.



Figur 23: Grafene fra eksempladata som ble brukt til å kontrollere formlene som blir brukt i utregningen av hastighet og strekning.

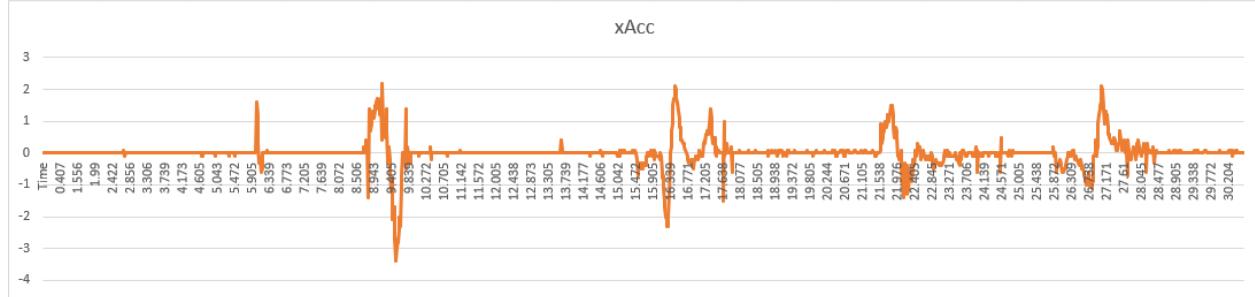
Hvis enheten er i ro har den en positiv kraft i negativ z-retning på grunn av tyngdekraften. Når man henter akselerasjonsdataen fra akselerometeret ved å bruke "linnear Acceleration" så skal tyngdekraften være filtrert bort. Denne filtreringen er ikke feilfri og man får en synlig påvirkning på akselerasjonsdataen i aksen som peker oppover. Ved å rotore enheten slik at positiv x-akse peker oppover mottar vi dataen i figur 24. Her ser vi at det meste av påvirkningen fra tyngdekraften kan filtreres vekk ved å trekke fra $0,2m/s^2$ i alle følgende

målinger. Denne verdien bestemmes gjennom en sjekk av de første dataen som kommer mens enheten ligger i ro. Dette gjøres under kalibreringen.



Figur 24: Ved å rottere enheten ser vi at gravitasjonen påvirker enheten positivt oppover på grunn av en filtrert akselerasjon

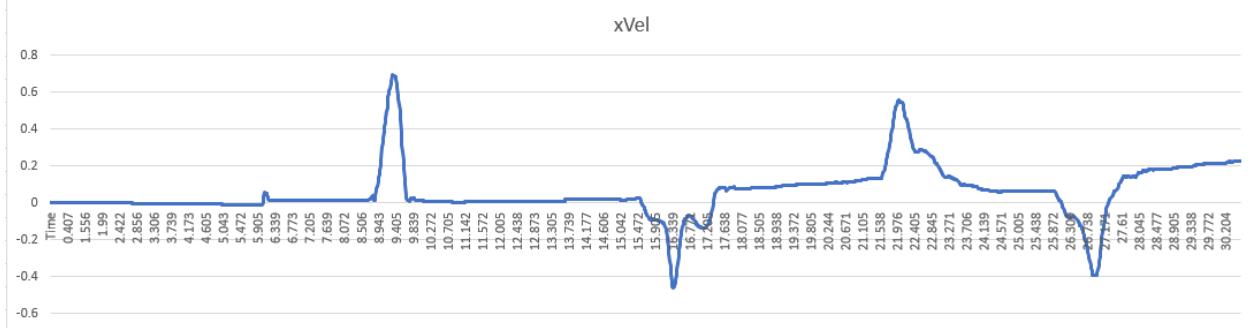
Ved å flytte enheten fram og tilbake mellom sammepunkt skal strekningen tilbakelagt være tilnærmet null. En lengre test viser tydelige feil i målingene. Testen varte over 30 sekund og enheten ble flyttet fram og tilbake to ganger. Testen foregikk uten bruk av filtre.



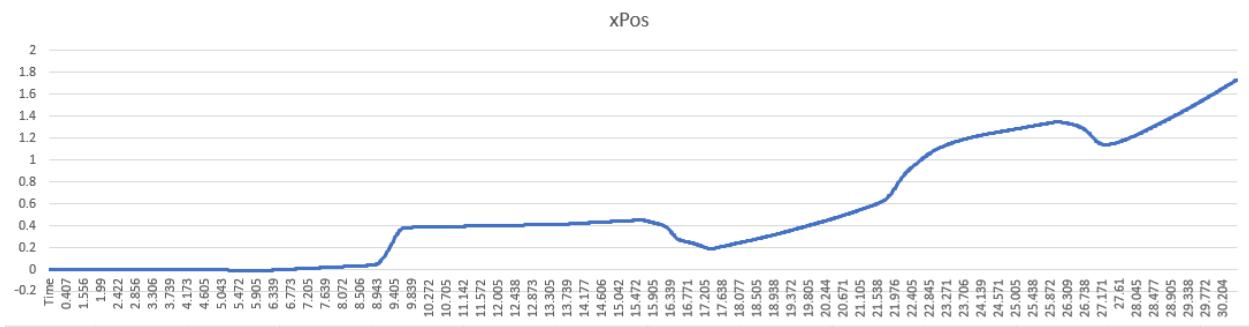
Figur 25: Akselerasjonsdata fra test i 30 sekunder uten filtere.

På grafen over akselerasjonsdataen i figur 25 ser man at det er bevegelse selv om enheten ligger i ro. Dette er støy som fjernes ved å sjekke om akselerasjonen er under et satt nivå. Dette vil over lang tid påvirke akselerasjonen negativt og gjør at strekningen vil bli kortere ettersom vi ikke har noen metode for å skille mellom reell akselerasjon i samme størrelse og fjerner både støy og reell akselerasjon innenfor samme område.

I Figur 25 ser man at grafen er veldig hakkete og ekstremverdiene er ekstra store. Dette er et typisk for et signal med lavfrekvensstøy. Da vi ikke mottar målingene som frekvensverdier vil vi heller prøve å flate ut de toppunktene og bunnpunktene. Dette kan gjøres ved å regne ut gjennomsnittet av de siste målingen erstattet de ekstreme verdiene med gjennomsnittet. Dette vil gi en jevnere kurve, men kan føre til at deler av reell måling også jevnes ut og det blir en mulig feilkilde. Det er allikevel en god erstatning for lowpass-filteret.



Figur 26: Hastigheten over tid fra test i 30 sekunder uten filtere.

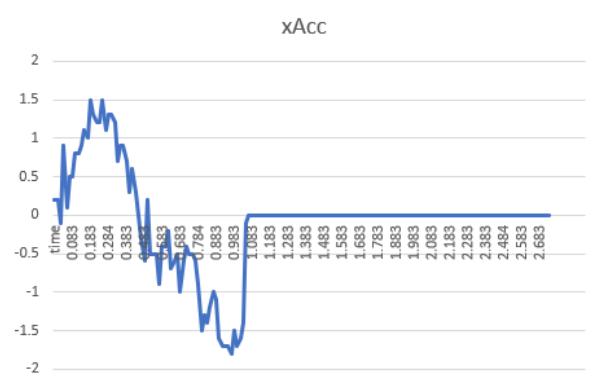


Figur 27: Strekningen over tid fra test i 30 sekunder uten filtere.

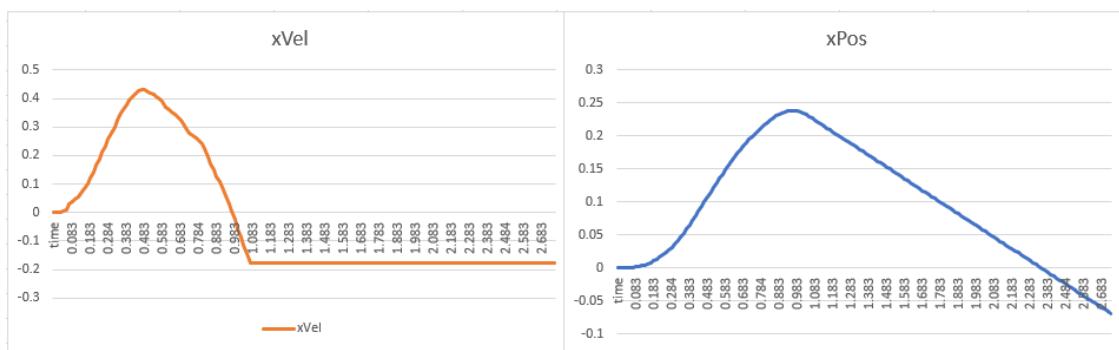
Grafen med data fra den utregnede hastigheten i figur 26 viser en økende positiv hastighet selv når akselerasjonen er null og enheten ligger i ro. Dette skyldes både støyen og at akselerasjonsdataen ikke utlignes 100% etter avsluttet bevegelse. Da blir hastigheten positiv selv om akselerasjonen er null. Dette kan korrigeres ved å sjekke om akselerasjonen har vært null over flere målinger. Dersom akselerasjonen er null over lengre tid settes hastigheten til null. Dette kan gjøres med en ZeroCounter.

Når enheten ligger i ro mellom bevegelse burde grafen med dataen over posisjonsendringene være flat. Ettersom hastigheten ikke bli null har streknings-grafen i stedet en økende positiv endring ettersom hastigheten øker.

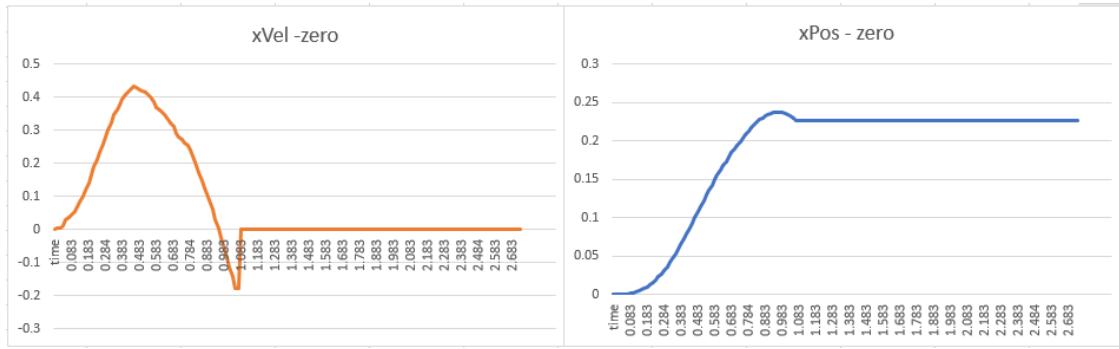
Ved å se på et utsnitt av en akselerasjonstest hvor akselerasjonen er null over tid, men den positive og negative akselerasjonen er ikke like stor. Her ser man i figur 29 at selvom akselerasjonen forblir null ettter 1.083 sekunder så forblir hastigheten negativ. Den konstant negative hastigheten påvirker strekningen negativt selv om strekning burde hold seg lik. I figur 30 er settet zeroCounteren hastigheten til null etter noen målinger som gjør resultatet mer reelt da enheten ligger i ro og ikke beveger seg i negativ retning.



Figur 28: Akselerasjonsdata over tid med ZeroCounter



Figur 29: Hastighet og strekning ut i fra data i figur 28



Figur 30: Hastighet og strekning med påvirkning fra ZeroCounter

3.4 3D-grafikk

3D miljøet i applikasjonen er et av de definerende kravene til en AR-applikasjon beskrevet i kapittel 1.2. Til dette trengs det en renderer og en scene med objekter som kan rendres ut.

3.4.1 Three.js

Three.js modulen lastes ned fra git-repoet eller Threejs.org[55] og importeres i modulen hvor biblioteket skal benyttes. 3D-miljøet bygges opp ved bruk av objekter som får tildelt en geometrisk utforming og et materiale. Objektene legges til i scene objektet. Kameraet er også et objekt som tilhører scenen. Kamera-objektet bestemmer hvilke bilder som rendres ut fra scene-objektet.



Figur 31: En scene med en 3D-modell og et perspektivisk kamera.

Før scenen kan rendres ut i nettleseren må WebGL-renderen hentes inn fra Three-APIet. Renderen har en egen metode hvor man sender inn scenen man ønsker rendret og kameraet som bestemmer utsnittet. Dersom scenen skal animeres eller ha bevegelse lages det en funksjon som påkaller render-funksjonen før hver nye frame skal tegnes. Window-objektet inneholder metoden “requestAnimationFrame()”[59]. Denne påkaller en valgfri funksjon før den tegner opp scenen på nytt. Ved å påkalle en funksjon som oppdaterer scenen før den rendres ut, kan funksjonen benyttes til interaktiv funksjonalitet.

3.4.2 Sensordata i 3D-scenen

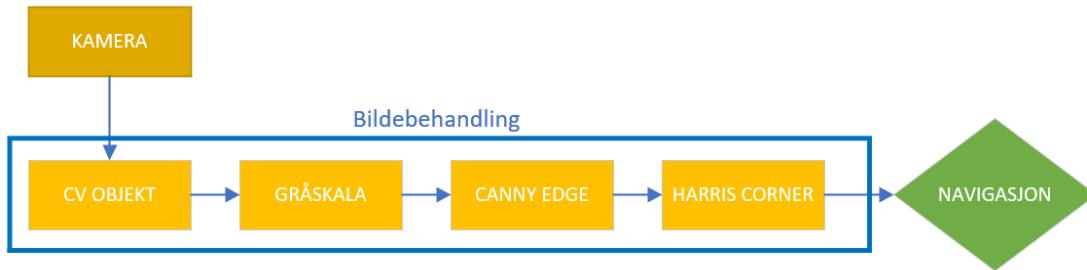
Magnetometeret gir, som beskrevet i kapittel 2.5.3, data i sanntid om enhetens orientasjon. Denne informasjonen kan overføres til det visuelle kamera-objektet i 3D-scenen. Ved at kameraet i 3D-scenen beveger seg samtidig med enheten vil det øke opplevelsen av at objektene befinner seg i det fysiske rommet, men er kun synlig gjennom enhetens skerm.

Dataen fra magnetometeret må først regnes om til radianer. Deretter kan dataen implemeteres direkte i kameraet rotasjonsverdier som her ligger i x, y og z. Ettersom enhetens koordinatsystem og Three-kameraets koordinatsystem ikke er likt orientert utføres det *Euler transformasjon* av dataen. Etter dette oppdateres orientasjonen til det visuelle kameraet hver gang dataen fra magnetometeret oppdateres. I samme modul utføres det en sjekk på om telefonen er i portrett eller landskapsmodus, i tillegg til at det tas hensyn til at applikasjonen benytter seg av kameraet som befinner seg på baksiden av enheten. Dette korrigeres ved å rotere negativt med $\pi/2$ om x-aksen.

Skalering av 3D-modeller etter omgivelsene kan gjøres ved hjelp av en dybdesensor eller som forklart i kapittel 3.5.1 - ved hjelp av akselerasjonen fra akselerometeret, informasjon om kameraet og et punkt i rommet. Gjenkjennelige punkt i rommet som kan detekteres gjennom bildebehandling kan brukes til å feste modellen til rommet.

3.5 Bildebehandling

Som beskrevet i kapittel 2.7.1 så kan det benyttes forskjellige metoder for å gjenkjenne kanter, hjørner og objekter i et bilde. Innledningsvis i figur 3 ser man tydelig hvor bildebehandling er implementert i applikasjonen. Kamera står er hovedkilden til dataen som benyttes bildebehandlingen og vil videre kunne være med på å påvirke navigasjon i 3D modulen. I oppgaven er det valgt å benytte metoden *HarrisCorner Detection* for å detektere hjørner i bildet. Strukturen for hvordan dette er implementert i koden sees i figur 32.

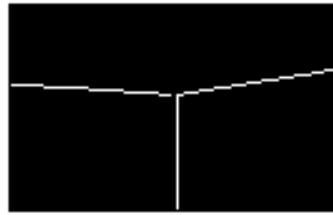


Figur 32: Struktur på bildebehandling

Både *HarrisCorner* og *CannyEdge* må ha et gråskala bilde som input, første steg er derfor å konvertere bildet kamera gir til riktig format. OpenCV funksjonen *cvtColor()* løser dette hvor argument *COLOR_BGR2GRAY* blir benyttet for å transformere bilde i farge over til et bilde i gråskala. Dette bildet blir så sendt inn i *CannyEdge* som igjen gir ut et bilde med kun sorte eller hvite piksler, hvor da de hvite pikslene i bildet er resultatet av de detekterte kantene (se Figur 34). Parameterne i *CannyEdge* funksjonen er justert etter et mobilkamera i et godt belyst rom. Bildet som kommer ut av *CannyEdge* funksjonen blir så sendt inn i *HarrisCorner* som igjen gir ut et bilde i sort og hvit. Hvite piksler vil da tilsvare alle hjørnene som er detektert (se Figur 35). Det ble brukt samme metode for å justere parameterne til *HarrisCorner* som i *CannyEdge*.



Figur 33: Bildet av et hjørne i farger



Figur 34: Bildet etter det er kjørt igjennom *CannyEdge*



Figur 35: Bildet etter det er kjørt igjennom *HarrisCorner*

Dataen som kommer ut av bildebehandling vil være piksel-koordinat til det detekterte hjørnet. Videre vil dette bli brukt som et ankerpunkt hvor 3D objekter kan festes til.

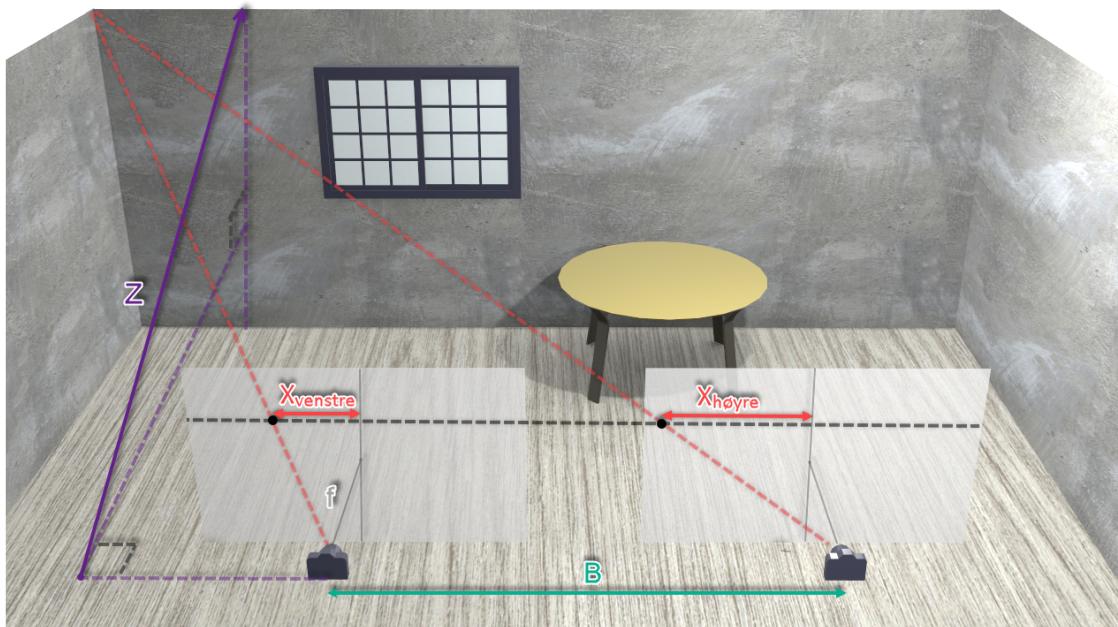
For å bruke punktene som ankerpunkt må et enkeltpunkt velges og lagres. For å velge ut et spesifikt punkt som skal lagres kan brukeren velge et område gjennom et trykk. Det blir så opprettet et mindre canvas-element med utgangspunkt i de valgte koordinatene.

Gjennom metoden “`.drawImage`” til *Canvas 2D APIet* kan en kopi av videoen klippes ut og tegnes i det mindre canvaset. Dette gir mulighet til å gjennomføre detektering i kun det lille canvaset basert på det dataen i det originale bildet. Når hjørnets koordinat er detektert flyttes canvaset slik at det detekterte hjørnet befinner seg i midten av canvaset. Canvaset sin posisjon blir lagret i et globalt objekt. Da kan hjørnets forrige posisjon benyttes til å lete etter hjørnet i tilnærmet samme posisjon neste gang applikasjonen går igjennom EdgeDetection-funksjonen. Så lenge detekterings-funksjonene i canvaset finner et hjørne innenfor området sitt vil den holde på hjørnets posisjon i bildet. Hvis hjørnet havner på utsiden av canvaset ved store bevegelser i enheten, vil detekteringen ikke klare finne det igjen og hjørnet er mistet. For at canvaset skal klare å flytte seg med det detekterte hjørnet er man avhengig av at funksjonen kjøres ofte og fort nok til å følge enheten bevegelser. At detekteringen ikke trenger å foregå på det totale bildet konstant kan spare prosessorkraft.

En samling av data for flere av hjørnene i bildet vil bli brukt til å danne bilde om skalering av objekter i tillegg til en dybdeberegning. Dette er nærmere forklart i kapittel 3.5.1.

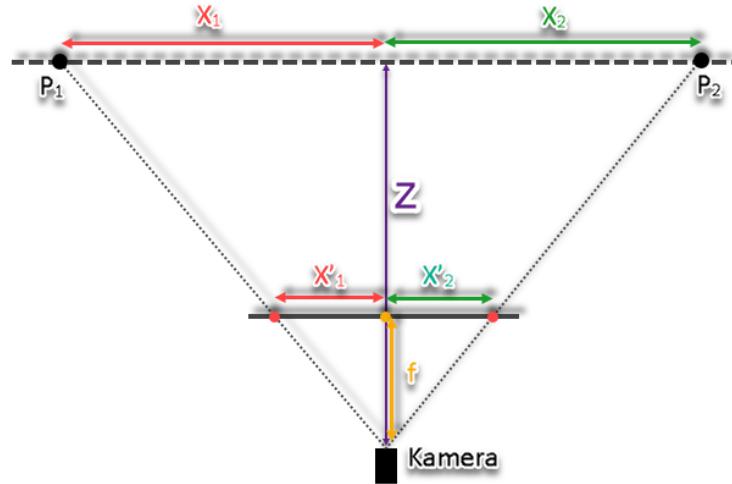
3.5.1 Dybdeberegning

For å kunne gjennomføre en dybdeberegning med kun ett kamera er det nødvendig å kalibrere opp kameraet. Det er valgt å bruke en litt utradisjonell metode hvor kamera hele tiden må ha minst 2 punkter som har et fast forhold til hverandre, som for eksempel to hjørner av samme vegg. Når da dybdekalibrering ligger i bunn kan en til enhver tid si noe om kamera har beveget seg nærmere eller lengre ifra et objekt, og 3D objekter kan da skaleres etter dette. Metoden som benyttes for å kalibrere er beskrevet i kapittel 2.7.2.



Figur 36: 3D skisse av projeksjon av detektert hjørner inn på bildeplanet

I Figur 36 er det skissert opp tilsvarende figur som en kan finne i kapittel 2.7.2, bare i 3D. Ut ifra formelen for dybdeberegning finner vi avstanden Z . Avstanden B beregnes ut ifra dobbeltintegralet av akselerasjondata. Forklaring og matematisk beregning for lengdemålingen er forklart i kapittel 2.5.3.



Figur 37: Skisse av 2 punkter som er projektert i bildeplanet

Punktene P_1 og P_2 som er detektert fra bildebehandling kan sees i Figur 37. For å beregne den fysiske avstanden mellom punktene er det tatt i bruk formlikhet. Den totale lengden vil være summen av X_1 og X_2 , og fokuseres det kun på punktet P_1 kan formlikheten beskrives slik:

$$\frac{x'_1}{f} = \frac{X_1}{Z} \Rightarrow X_1 = \frac{x'_1}{f} \cdot Z$$

Det samme gjelder for punktet P_2 og den totale lengden mellom punktene kan derfor uttrykkes slik:

$$X_1 + X_2 = \frac{x'_1 + x'_2}{f} \cdot Z$$

Når en vet lengden mellom P_1 og P_2 både i bildeplanet og i virkeligheten kan alle andre avstander også beregnes med samme forhold. Vi kan dermed si at *skaleringsfaktor* er avstand i bildeplanet (D_b) delt på avstand i virkeligheten (D_v), altså D_b/D_v . Det betyr at vi hvis vi vet avstanden i piksler og finner vi avstanden i virkeligheten.

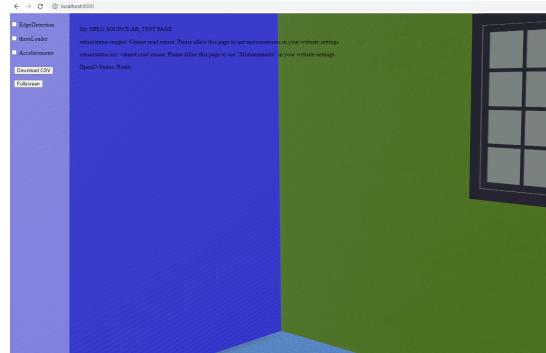
4 Resultat

I dette kapittel vises resultatet av arbeidet med prosjektet. Det innebærer resultatetene fra metodene benyttet i kapittel 3.

Resultatet av bachelorprosjektet er en applikasjon som kombinerer 3D-grafikk med bilder fra enhetens kamera. Data fra sensorer blir brukt til å rotere kamera-objektet og i dybdeberegning. Applikasjonen er bygget utelukkende ved hjelp av openSource-biblioteker som gir tilgang til å bruke, distribuere og videreutvikle applikasjonen. Applikasjonen kjøres som en webapplikasjon som gjør den tilgjengelig for enheter gjennom netteleseren uavhengig av operativsystemet.

4.1 Applikasjonsdesign

Applikasjon er bygget for å kunne teste moduler og funksjoner. Det er derfor lagt vekt på at det skal være mulig å lese data rett på skjermen og skru av og på tunge prosesser underveis. Utseende har bare blitt vektlagt i tilfeller hvor det var nødvendig for utføring av testene. Applikasjonen kjører som en webapplikasjon gjennom Heroku og GitHub og er derfor uavhengig av operativsystemet og tilgjengelig lagringsplass på enheten.



Figur 38: Skjermbilde av applikasjonen under oppstart i localhost.

Index.html: Html-filen inneholder alle DOM elemter, henter inn script og moduler, registrerer klikk og printer ut bilder og tekst med informasjon.

Style.css: Dette er en fil som beskriver hvordan HTMLen skal utformes. Faktorene som påvirker resultatet av applikasjonen mest er muligheten til å sette en fixedposisjon på elementene for at de skal legges seg lagvis, og justering av bredde og høyde for optimal utnyttelse av skjermen.

Main.js: Scriptets oppretter eventlistener for deviceorientation(magnetometeret) og devicemotion(akselerometeret) samt inneholder funksjoner som kommuniserer direkte med index.html, formatering av tall og visning av informasjon i tekstformat på skjermen.

Andre script og moduler som er relevant for prosjektet:

- **CameraLoader.js:** Henter stream fra enhetens kamera og sender den til video-elementet i index.html.
- **ThreeLoader.js:** Bygger opp 3D-scenen med renderer og oppdaterer scenen underveis.
- **HandleOrientation.js:** Henter data fra magnetometeret og regner om til radianer.
- **HandleMotion.js:** Henter data fra fra akselerometeret, regner ut hastigheten og avstand fra oppstartsposisjonen.
- **OpenCv.js:** Inneholder OpenCv-modul tilpasset javascript.
- **DeviceOrientationControls.js:** Formaterer orientasjons-data mottatt fra magnetometeret til koordinater som kan brukes i Three-scenen og opp.

4.2 Navigasjon og bevegelse

På skjermen er det muligheter for å benytte berøring eller trykk for å sende koordinater inn i modulene. Det er også mulig å slå av noen funksjonaliteter eller lag for å teste applikasjonen med og uten deler av modulene. Dette gjør det enklere å kunne teste funksjonene og modulene under utvikling.

Sensordataen fra magnetometeret kjøres rett inn i 3D-scenen og påvirker orientasjonen til kamera-objektet. Dette blir en del av den interaktive påvirkningen av 3D-miljøet i sanntid. Det ble også testet om det var mulig å benytte dataen fra akselerometeret til å forflytte kameraet horisontalt langs x- og y- aksene, men dataen hadde mye støy. Driften og feilkildene i utregningene ble altfor store til å fungere godt over tid. Se testresultatene i kapittel 4.3

Ettersom målingen fra akselerometeret fungerer best i over kortere tidsrom ble det også testet mulighetene for å benytte målingen til å finne en skaleringsfaktor - mer om det i kapittel 4.5.1.

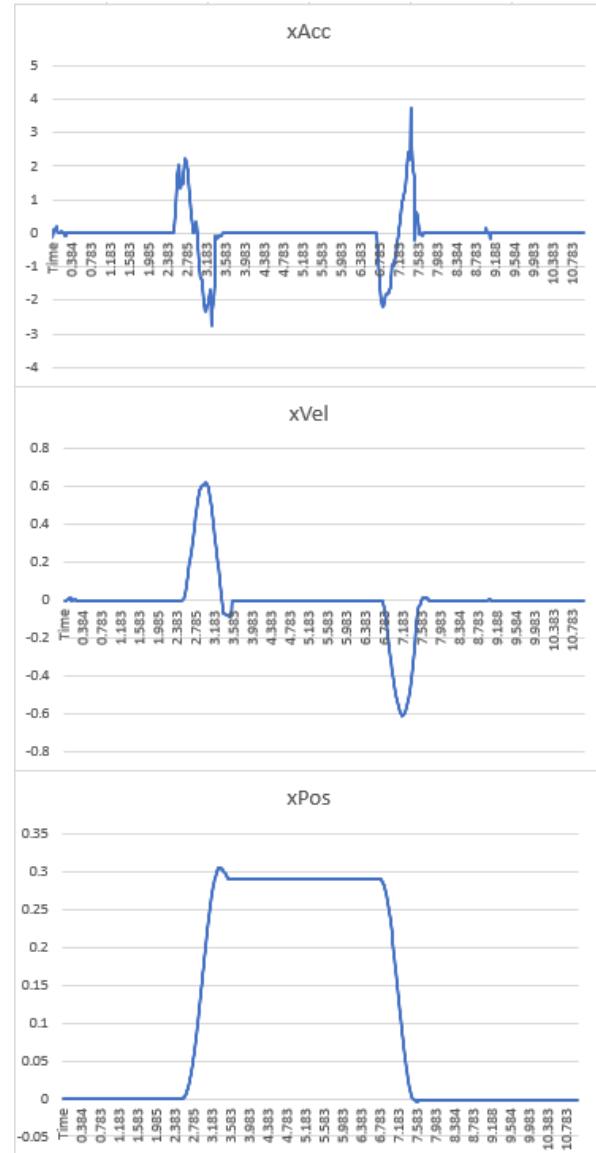
4.2.1 Magnetometer

Sensordataen fra magnetometeret forteller om enhetens orientering ut ifra jordens magnetfelt. Informasjonen brukes til å rotere kameraet-objektet i 3D-miljøet med tilsvarende rotasjon som enheten. Orienteringen fungerer ved hjelp av Euler omregningen til verdier som er tilpasset Three-objektets koordinatsystem.

4.3 Akselerometer

Dataen fra akselerometeret blir brukt til å beregne horisontal bevegelse langs aksene. Ved bruk av dobbel integrasjon og filtre måles avstanden fra punktet hvor applikasjonen startes. Over tid blir avstanden mellom den utregnede avstanden og den reelle avstanden altfor stor til å kunne benyttes til navigering.

Grafen i figur 39 viser akselerasjonen, hastigheten og strekningen målt under en horisontal bevegelse langs x-aksen på 30 cm. Her blir det brukt kalibrering, fjerning av støy, lowpass-filter og ZeroCounter. På det meste viser strekningsmålingen at avstanden er 30cm, men flyter litt tilbake. Etter at enheten blir flyttet like mye tilbake er resultatet ganske nøyaktig 0cm fra start som forventet. Konklusjonen fra testene av akselerasjonsdataen fra akselerometeret(kapittel 3.3.3) er at med riktig filtrering av dataen kan enhetens akselerasjon benyttes til å regne ut en omtrentlig strekning. Det krever kontrollerte bevegelser av brukeren, rask databehandling for raske målinger, at det ikke måles over for langt tidsrom, en kalibrering for å unngå feil fra tyngdekraften og at det blir benyttet riktige filtre som kan fjerne deler av støyen uten å fjerne for mye av nødvendig data.



Figur 39: Resultatet av en akselerasjonsmåling med tilpassede filtre

4.3.1 Andre bruksområder

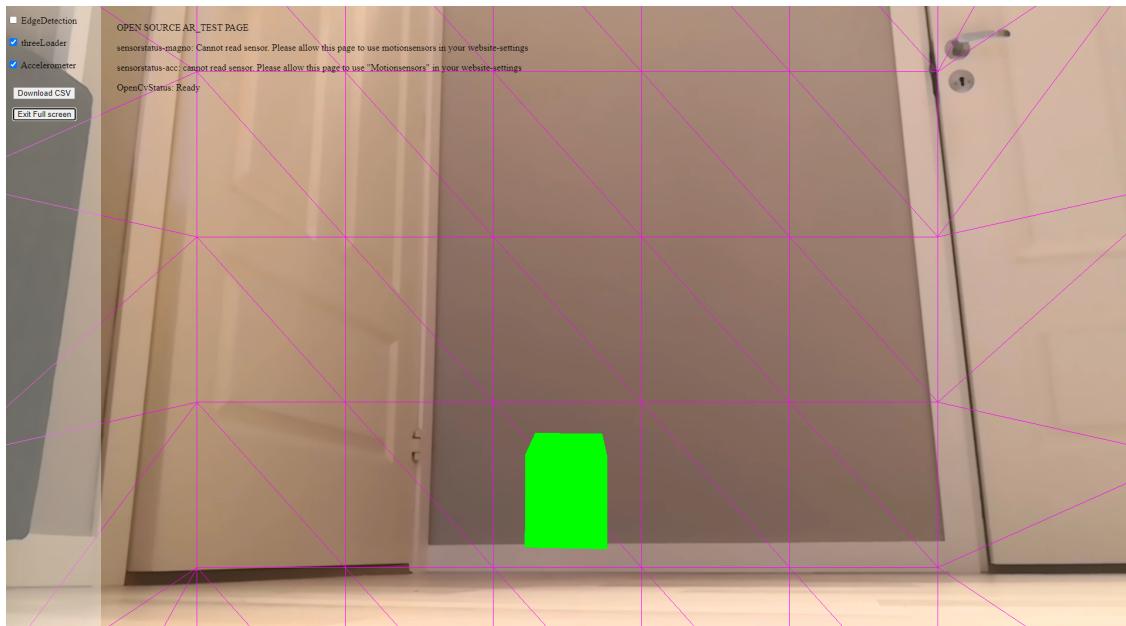
Fra akselerasjonstesten[3.3.3] viser det seg at dataen fra akselerometeret har mange feilkilder og egner seg ikke til navigasjon over lengre tid, men funksjoner som kun krever en måling over kortere tid kan allikevel være aktuelle bruksområder. For eksempel kan den utregnede strekningen være nyttig under dybdeberegning[3.5.1].

4.4 3D-grafikk

Det ble bygget en Three-scene med kamera-objekt, en kube og hjelpe linjer. Hjelpe linjene benyttes som veger som er mulig å se kamerabilde igjennom. Det var tenkt at de skulle flyttes etter dybdeberegningene som gir en avstand fra veggen, men ettersom dybdeberegningene ikke kunne bli utført har den kun satt en avstand fra oppstart.

Kameraobjektet blir plassert i Three-scenen og mottar formatert orienteringsdata fra magnetometeret. Kuben blir plassert ut i scenen med en konstant avstand og posisjon i forhold til kamera-objektet. Når kamera-objektet orienteres ved hjelp av bevegelsene i enheten vil kuben holde en relativt fast posisjon i forhold til virkeligheten.

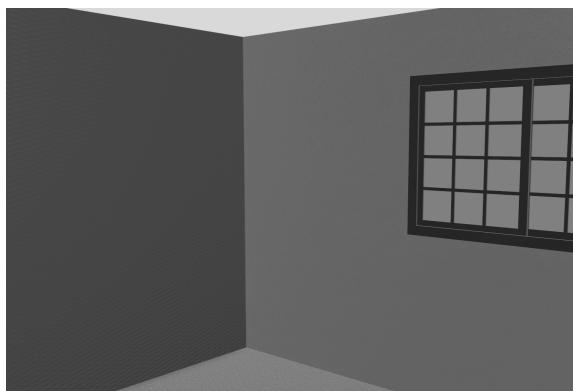
Ved hjelp av ankerpunkter fra bildebehandlingen(kapittel 4.5) er det mulig å feste objektet til faste punkter i rommet, men ankerpunktene under testene var ikke stabile nok til at dette ble implementert i resultatet.



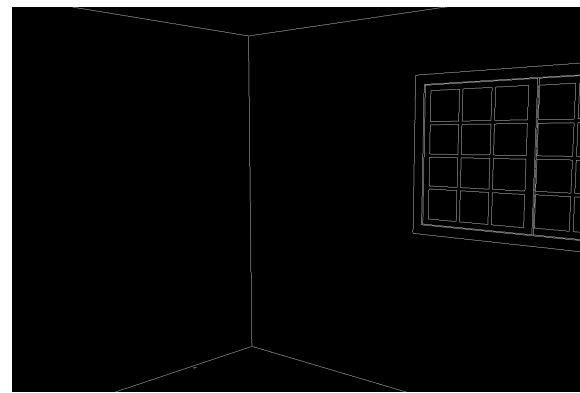
Figur 40: Three-scenen med gjennomsiktig 3D mesh (lilla) og en grønn 3D boks

4.5 Bildebehandling

Resultatet av hjørne-deteksjonen er vist i sekvens i figur 41, 42, 43 og 44. Resultatet er vist fra et virtuelt kamera i et simulert 3D rom da et rom i virkeligheten med slette vegger og hjørner ikke er lett tilgjengelig. Først ser en bildet som er tatt av kamera gjort om i gråskala bilde. Figur 42 er så kjørt igjennom *CannyEdge* og viser alle detekterte kanter som hvite streker. I figur 43 er bildet fra *CannyEdge* kjørt igjennom *HarrisCorner* og viser alle detekterte hjørner som hvite "prikker". Figur 44 viser til slutt resultatet av *HarrisCorner* bildet som kjøres igjennom et *erode*- og *dilate*-filter. De detekterte hjørnene er her enda mer synliggjort og en del "falske" hjørner er filtrert bort.



Figur 41: Kamera sitt bilde i gråskala bilde



Figur 42: Gråskala bildet kjørt igjennom *CannyEdge*

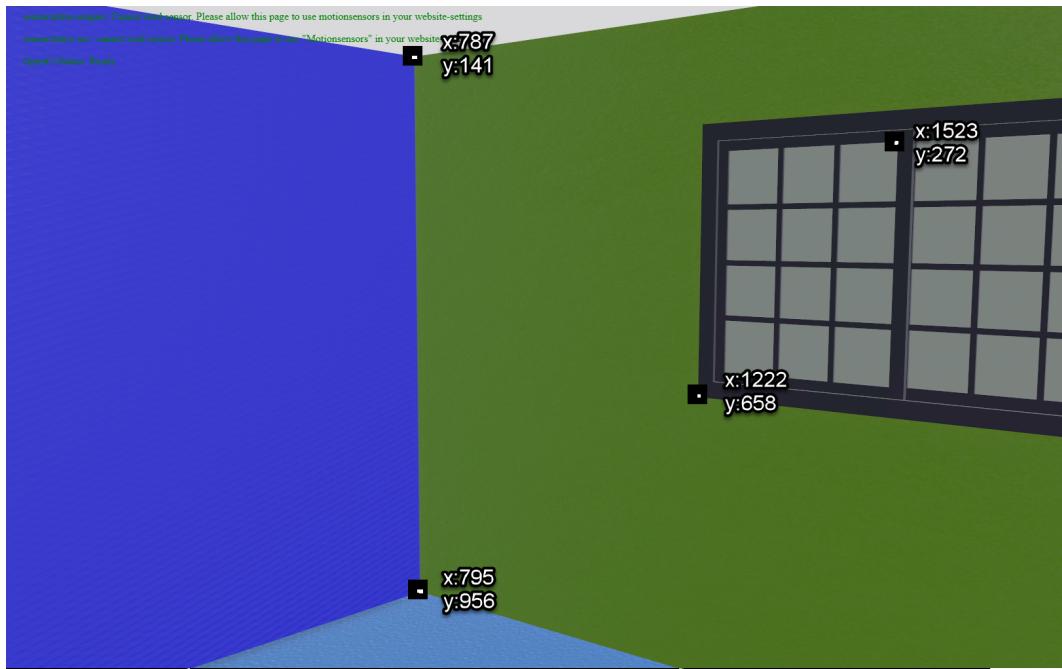


Figur 43: *CannyEdge* bildet kjørt igjennom *HarrisCorner*



Figur 44: *HarrisCorner* bildet kjørt igjennom *erode* og *dilate* filter

I figur 45 vises resultatet av 4 canvas, valgt gjennom berøring, som tracker hjørnene. Brukeren klikker i nærheten av hjørnet som er ønsket å benytte. Det utføres filtrering og behandling av matrisene i de oppgitte områdene og de detekterte hjørnene vises som en liten hvit prikk. Siden de detekterte hjørnene er i bildeplanet så kan alle pikselkoordinatene til hvert av hjørnene hentes ut slik det er vist i figuren.



Figur 45: Bildet av et rom hvor 4 hjørner er valgt og tracket

4.5.1 Dybdeberegning

For å kunne utføre dybdeberegningen som beskrevet i kapittel 3.5.1 trengs det to stk bilder med koordinatene til samme punkt fra begge bildene. Avstanden mellom de to bildene som bli tatt og fokallengden til kameraet. Ettersom man får hentet ut koordinatene til punktet i piksler trenger man størrelsesforholdet til kameraet (s_c) i m/s^2 for å kunne regne ut avstanden i mm. Det lyktes ikke i å finne dette størrelsesforholdet til enhetens kamerasensor. Men faktoren kan regnes ut manuelt ved å benytte en kjent dybde/avstandsmåling. Dette gjør at funksjonen ikke kan benyttes uten at brukeren manuelt gir applikasjonen en kjent avstand til punktene som trackes.

Dersom det gjøres mulig å regne ut den fysiske avstanden til punkter på et bilde er det mulig å regne ut den fysiske avstanden mellom to punkter som vises i bildeplanet. Dette kan brukes til skalering av objekter ettersom skaleringen kan følge forholdet mellom den fysiske avstanden og avstanden i piksler i bildeplanet.

5 Diskusjon

En Augmented Reality-applikasjon som oppfyller de definerende kravene inneholder et 3D-miljø med virtuelle elementer og live-bilder fra virkeligheten i bakgrunnen. 3D objektene påvirkes av innebygde sensorer som gjør at 3D-miljøet blir interaktivt. Resultatet av prosjektarbeidet oppfyller de definerende kravene ved hjelp av lagvis visning av Three.js-scenen og bilde fra kameraet. Det visuelle kamera-objektet blir påvirket av data fra orienteringssensoren magnetometeret.

Det har kun blitt brukt OpenSource-biblioteker hvor det henvises i koden til forfatteren av biblioteket der det ble ønsket. Web-applikasjonen gjør det mulig å kjøre applikasjonen på ulike enheter, det har allikevel ikke blitt testet skikkelig hvilke enheter som støtter sensor API som er benyttet. Det er allikevel bevisst blitt valgt bort APIer som ble oppgitt å være kompatibel med et fåtall av nettlesere og funnet alternativer som skal være kompatible med de fleste.

Selv om valgte løsningsmetoder oppfyller minstekravene til en Augmented Reality-applikasjon fungerer ikke alle løsningene optimalt. Ved å benytte en bedre tracking eller detection metode kan 3D-objektene som er festet til virkeligheten gi et enda mer realistisk opplevelse. Metoder som kan overføre lyssetting fra virkelighet og over til 3D objekter i rommet vil også gjøre opplevelsen mer realistisk. Ingen slike løsninger ble testet ut i dette prosjektet.

Data fra magnetometeret fungerte til å orientere kamera-objektet i Three-scenen etter enhetens orientasjon. Resterende objekter var dermed ikke avhengig av å ha synlige markører i bildet for å beholde posisjonen i Three-scenen. Ettersom 3D-objektene ikke har dybdeskalering eller ankerfeste basert på omgivelsene oppleves objektene fremdeles som svevende. De blir ikke en naturlig del av omgivelsene da de ikke responderer likt som andre objekter i miljøet på orientasjonsendringene grunnet manglende skalering.

Akselerasjonsdataen fra akselerometeret fungerer til å måle opp strekning langs aksene over et kortere tidsrom. Den kan grunnet større avvik over lengre perioder ikke benyttes til navigasjon langs aksene. Det går allikevel an å teste videre om det er mulig å rekalibrere dataen etter et fast tidsintervall for å teste om dette kan bidra til å redusere avvik over tid. Akselerometerdataen kan allikevel vurderes som nøyaktig nok til å utføre dybdeberegninger og reelle avstandsmålinger gjennom kameraet dersom det ikke er behov for nøyaktige målinger i applikasjonen.

Bildebehandlingen som bidro til å finne hjørner ved hjelp av enhetens kamera klarte å finne hjørner og piksel-koordinater. Dette krevde god, jevn belysning og lite forstyrrende elementer rundt hjørnene. Brukere befinner seg sjeldent i optimale lysforhold og helt tomme rom. Dette gjør at denne løsningen ikke er optimal alene til å finne ankerpunkter for 3D-grafikken eller punkter til dybdeberegning. For å forbedre funksjonen kan det testes med å legge på ulike filtere, teste andre filtreringsmetoder av bildene eller utvide søkerområdene størrelse. Gyroskopet benyttes i enhetenes stabilisatorer i kameraapplikasjoner, kanskje kan

det også hjelpe under bildebehandlig for å minske forstyrrende bevegelser i bildet. Ved å lagre data fra sensorene om enhetens orientering da ankerpunktet sist ble registrert er det en ekstra faktor som øker mulighetene for å finne igjen samme punkt.

Under testing av funksjonene ble flere moduler kjørt samtidig, dette krevde mye av prosessorene og applikasjonen ble hakkete og kjørte saktere. Dette førte til unøyaktig data fra sensorene og mye feil under bildebehandling. Ettersom det kun er laget en “tråd” og modulene må vente på hverandre stopper de litt opp når for mange prosesser kjøres samtidig. En mulig løsning er *multithreading*[60]. Multithreading gjør det mulig å kjøre flere uavhengige prosesser samtidig ved hjelp av de samme prosessorene.

Det mangler mange faktorer eller funksjoner for at applikasjonen skal fungere optimalt. Det er også mange metoder som ikke ble testet ut i dette prosjektet. Flere av modulene ble ikke testet i kombinasjon med andre moduler eller sensorer. En kombinasjon av gyroskop for stabilisering og registrering av orientasjonsdata fra magnetometeret inn i bildebehandlingen kan kanskje gjøre resultatene bedre og mer pålitelige. Den store mengden med uprøvde metoder gjør at det ikke kan konkluderes om det er mulig å bruke vanlige innebygde sensorer i en webapplikasjon for å bygge en tilgjengelig Augmented Reality-applikasjon. Men prosjektet danner allikevel et grunnlag for å tro at det kan være mulig å få til applikasjoner med noe AR-funksjonalitet.

6 Konklusjon

Målet for prosjektet var å tilegne seg kunnskap om Augmented Reality og funksjonalitet som trengs for å oppnå et forventet nivå av interaktivitet og samhandling mellom virkeligheten og en virtuell verden. En 3D-modul ble implementert for å kunne kombinere 3D-grafikken med data fra virkeligheten. Data fra innebygde sensorer bidrar til interaktivitet i 3D-miljøet gjennom bevegelse som følger enheten i sanntid. En ytterlig grad av samhandling mellom det virtuelle og virkeligheten ble forsøkt gjennom bildebehandling. Flere av metodene i bildebehandlingen trenger ytterlige tilpassninger for å kunnne bidra positiv til interaktivitet.

Programvaren er utelukkende bygget ved hjelp av åpne biblioteker og selve applikasjonen er tilgjengelig for enheter uavhengig av operativsystem gjennom enhetens egne nettleser.

Det er etter min personlige mening verd å jobbe videre med OpenSource og tilpasse applikasjonen til enheter som ikke er bygget med alle nødvendige sensorer for Augmented Reality. Augmented Reality baserer seg på at sensorene skal gjøre applikasjonen interaktiv og gjøre den spennende og intuitiv å bruke ved å kombinere det visuelle med omgivelsene. Dersom teknologien har mindre kompatibilitetsbegrensinger kan det resultere i enkeltmoduler som kan gjøre det enklere for bedrifter eller andre å ta i bruk Augmented Reality.

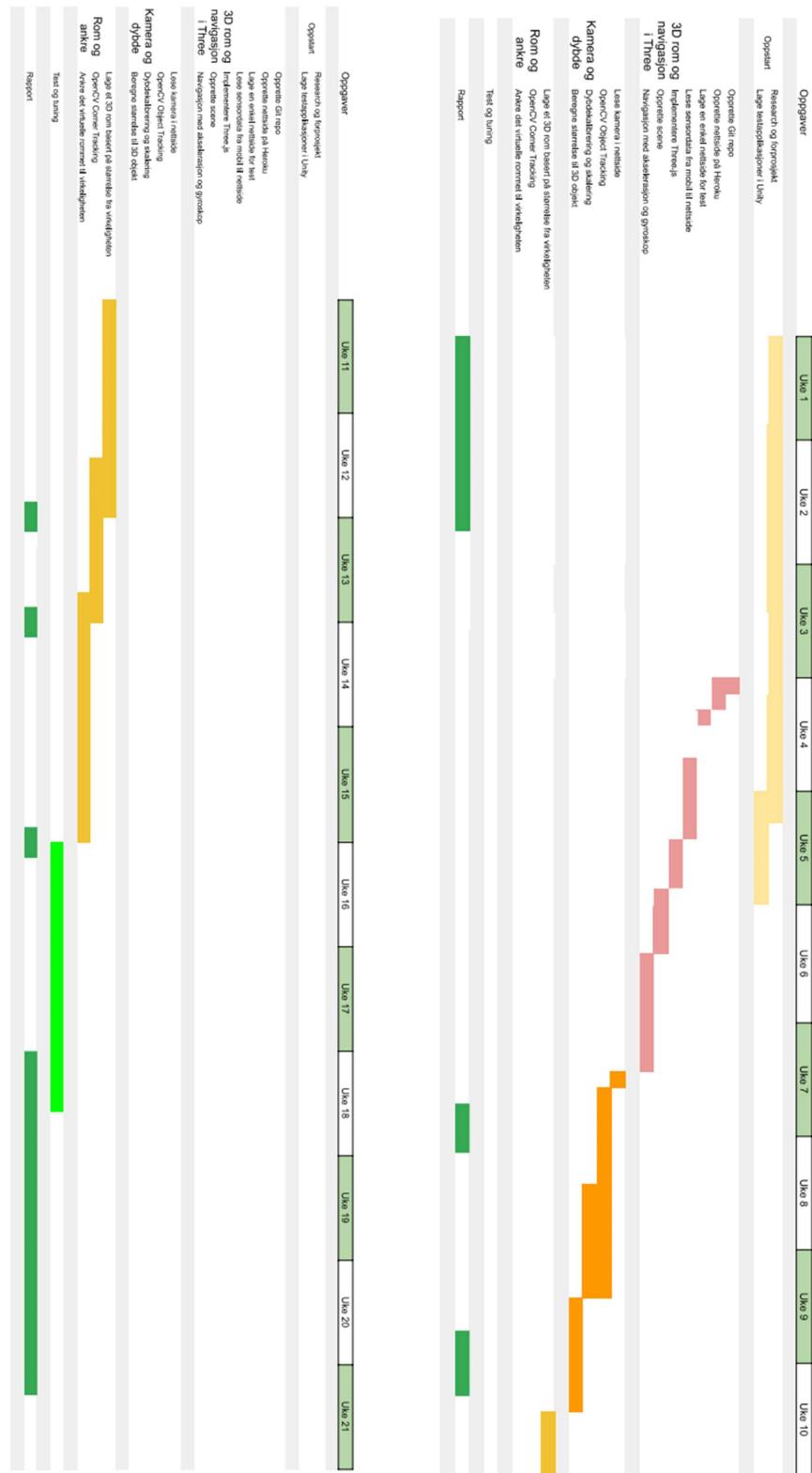
Referanser

- [1] Caroline Berg. Open source ar - testapplication . 2021.
<https://open-ar.herokuapp.com/>.
- [2] AR VR tech. Augmented reality vs. virtual reality vs. mixed reality. 2021.
<https://arvrtech.eu/>.
- [3] Kimberly Frazier Qian Wen, Harrison Ferrone and David Coulter. what is mixed reality? 2021. <https://docs.microsoft.com/>.
- [4] Apple. How does augmented reality work? . 2017.
<https://hbr.org/2017/>.
- [5] Veative. Why is gyroscope important for virtual reality? 2021. veative.com/blog/.
- [6] Android Developers. Motion sensors. 2020. source.android.com/.
- [7] Neha Jeswani. Demystifying huawei ml kit — face detection with react native. 2021.
medium.com/huawei-developers/.
- [8] Digit. Different types of augmented reality. 2021. digit.in/technology-guides/.
- [9] UIA Morten Ottestad. *MAS506 - Autonome Roboter*. UIA, 2018.
- [10] Unity Technologies. Unity real-time development platform. 2021. <https://unity.com/>.
- [11] Unreal Engine. The most powerful real-time 3d creation tool . 2021.
<https://www.unrealengine.com/en-US/>.
- [12] Vikingbad. Tegneprogram. 2021. <https://info.vikingbad.no/tegneprogram>.
- [13] Compusoft Group. Showcase 360. 2021. compusoftgroup.com/no.
- [14] IKEA presserom. Ikea place. 2017. ikea.com/no.
- [15] Google Developers. Build the feature . 2021.
<https://developers.google.com/ar>.
- [16] Apple. Utvidet virkelighet(augmented reality) . 2021.
apple.com/no/augmented-reality/.
- [17] MNX. Mems and nanotechnology applications. 2021. mems-exchange.org/MEMS/.
- [18] W3Schools. Javascript and html dom reference . 2021.
<https://www.w3schools.com/jsref/>.
- [19] THREE. three.js - documentation. 2021. <https://threejs.org/docs/>.
- [20] Unity Technologies. Unity ar foundation framework . 2021.
unity.com/unity/features/arfoundation.

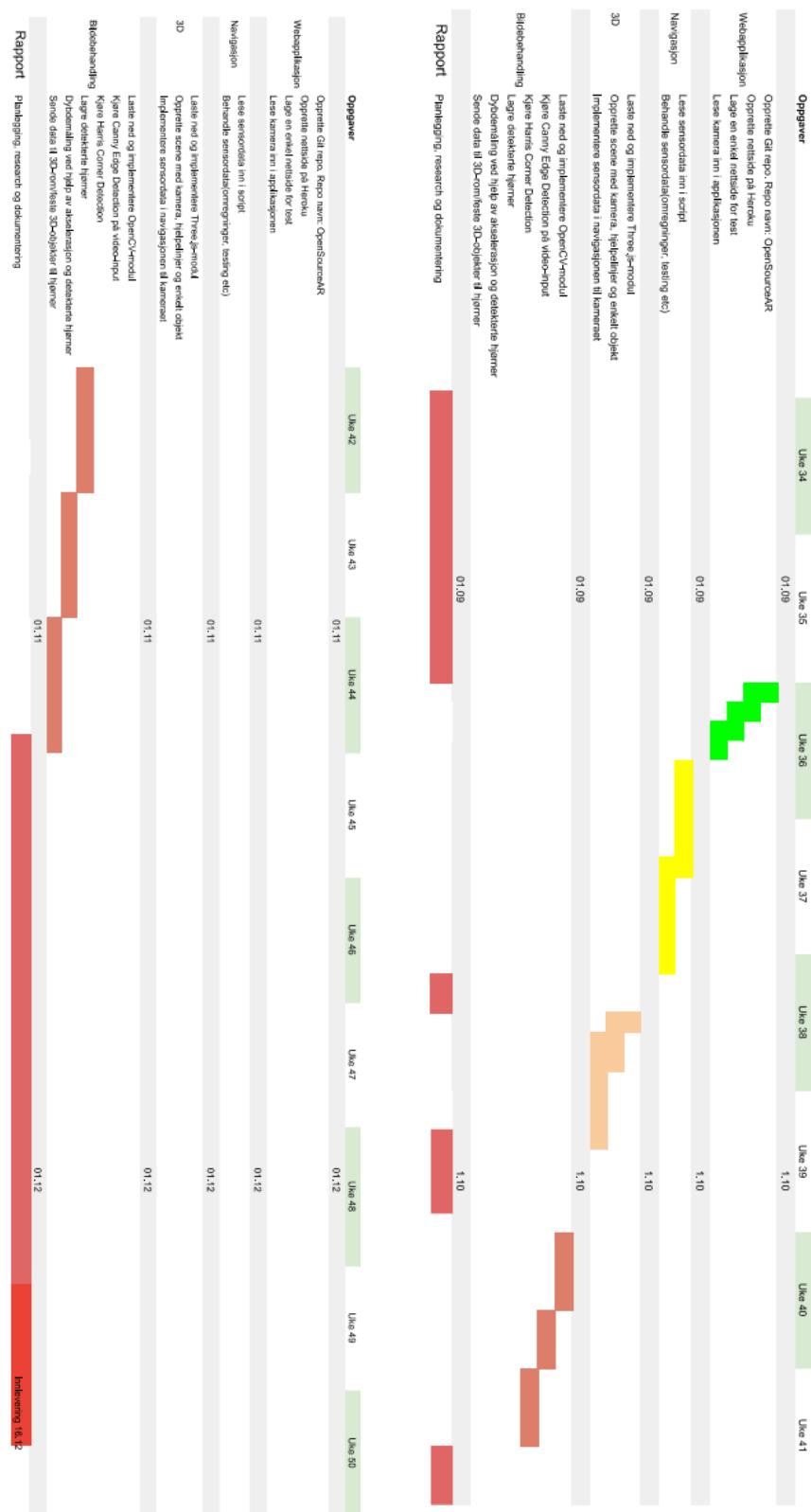
- [21] Youtube. Tutorials ar og unity3d - youtube . 2021. youtube.com/results?search_query=ar+unity3d.
- [22] learnopencv.com. Edge detection using opencv . 2021. learnopencv.com/edge-detection-using-opencv/.
- [23] Mozilla developer web api. 2021. developer.mozilla.org/en-US/docs/Web/API.
- [24] Visual studio code. <https://code.visualstudio.com/>.
- [25] OpenJS Foundation. About node.js. 2021. <https://nodejs.org/en/about/>.
- [26] Atlassian. Sourcetree. <https://www.sourcetreeapp.com/>.
- [27] Heroku. What is heroku. 2021. <https://www.heroku.com/what#summary>.
- [28] Dictionary.com. Extended reality. 2021. dictionary.com/browse/extended-reality.
- [29] Nancy Gupton og Patrick J. Kiger. What's the difference between ar, vr and mr? 2017. <https://www.fi.edu/difference-between-ar-vr-and-mr>.
- [30] The Pokémon Company. Pokémon go. 2021. <https://pokemongolive.com/>.
- [31] Greg Kipper and Joseph Rampolla. *Augmented Reality: an emerging technologies guide to AR*. Elsevier, 2012.
- [32] Google AR og VR. Google augmented reality, 2021. <https://arvr.google.com/ar/>.
- [33] Dhiraj Amin and Sharvari Govilkar. Comparative study of augmented reality sdks. *International Journal on Computational Science & Applications*, 5(1):11–26, 2015. [researchgate.net/](https://www.researchgate.net/).
- [34] ZealAR. About us. 2021. <https://zealar.com.au/about-us/>.
- [35] ZealAR. What is augmented reality and how does it work? 2019. zealar.com.au/.
- [36] James e. heppelmann. 2021. www.ptc.com/en/blog-authors/Jim-Heppelmann.
- [37] Ptc. 2021. <https://www.ptc.com/>.
- [38] PTC. Vuforia - ar platform). 2021. <https://www.ptc.com/en/products/vuforia>.
- [39] Microsoft. What is saas? 2021. <https://azure.microsoft.com/>.
- [40] Microsoft. What is paas? 2021. <https://azure.microsoft.com/>.
- [41] Microsoft. What is iaas? 2021. <https://azure.microsoft.com/>.
- [42] Microsoft. Serveless computing. 2021. <https://azure.microsoft.com/>.
- [43] Finn Jarle Kvalheim Tek.no. Mobilen din er full av sensorer.. 2014. www.tek.no/.

- [44] Andy Cho. What kind of sensors are embedded in smartphones). 2020. www.samsungsds.com/en/.
- [45] Android Developers. Accelerometer. 2020. source.android.com/devices/.
- [46] Android Developers. Gyroscope. 2020. source.android.com/devices/.
- [47] Android Developers. How does a gyroscope sensor work in your smartphone? 2020. techahedcorp.com/.
- [48] Android Developers. Magnetic field sensor. 2020. source.android.com/devices/.
- [49] W3C. Deviceorientation event specification . 2021. w3c.github.io/deviceorientation/.
- [50] Slant.co. What are the best javascript 3d frameworks? . 2021. <https://www.slant.co/topics/3658/>.
- [51] Dunebook. Top 21 javascript 3d library and frameworks . 2020. <https://www.dunebook.com/>.
- [52] Jake Rocheleau. The top 3d javascript libraries for web designers . 2018. 1stwebdesigner.com/3d-javascript-libraries/.
- [53] Openbase Experts. 10 best javascript 3d libraries . 2021. openbase.com/categories/js/.
- [54] Webgl. 2021. <https://www.khronos.org/webgl/>.
- [55] Threejs.org. 2021. <https://threejs.org/>.
- [56] Heroku Dev Center. Getting started on heroku with node.js . 2021. devcenter.heroku.com/.
- [57] Mozilla Developers. Sensor apis. 2021. developer.mozilla.org/.
- [58] Mozilla Developers. Permissions api. 2021. developer.mozilla.org/.
- [59] Mozilla Developers. Window.requestAnimationFrame() . 2021. developer.mozilla.org/.
- [60] Justin Stoltzfus. What is multithreading. 2021. techopedia.com/.

A Prosjektplan Våren 2021



B Prosjektplan Høsten 2021



C Møtereferater

Møtereferat - Valg av prosjektoppgave

Møtested: Zoom. Tidspunkt: 8.jan 2021 kl.9.00-9.30

Tilstede: Morgan Konnestad, Jostein Nordengen, Caroline Berg

Sammendrag

1. Valg av prosjekt i DAT303

På forhånd hadde Morgan og Jostein fått tilsendt en mail med ønske med forslag om et type AR-prosjekt. Prosjektets var av en så omfattende karakter at det kunne gjennomføres internt eller med en ekstern oppdragsgiver. VikingBad var kontaktet, men det var opp til studenten i hvilken grad hun ønsket å involvere dem som oppdragsgiver. Ventet fortsatt på svar fra representanter fra VikingBad AS.

Prosjektets elementer innebar bruk av AR-teknologi og bildebehandling for å ta mål, samt behandle 3D modeller i en applikasjon. Mer konkret prosjektbeskrivelse skulle arbeides fram sammen med eventuell oppdragsgiver i et møte og gjennom arbeidet med forprosjektet.

2. Gjennomgang av "Huskliste for DAT303"

- Studenten jobber i egen gruppe. Behøver ikke skriver "Gruppekontrakt".
- Studenten har gir beskjed til veiledere så snart rapporten for forprosjektet er levert (senest 26.februar). Det avtales da et nytt møte for å gå igjennom plan for videre arbeid.
- Pressemelding skrives normaltsett i mai og er derfor ikke prioritert enda
- Timeliste føres under alt arbeid med prosjektet inkludert møter med veiledere, møte med bedriften, research, rapportskriving og arbeid med selve produktet.
- Det skal skrives et kort møtereferat fra alle møter som gjennomføres med veiledere der alle beslutninger som tas skal komme fram.

3. Plan for videre arbeid

- Bestemme oppdragsgiver for prosjektet. Venter på svar fra VikingBad AS før avgjørelsen tas.
- Bestemme tid for møte når studenten hører tilbake fra VikingBad.
- Definere prosjektets rammer og betingelser. Dokumenteres i rapporten for forprosjektet.

Møtereferat - Oppstartsmøte med bedrift

Møtested: Teams. Tidspunkt: 14.jan 2021 kl.14.00-15.00

Tilstede: Morgan Konnestad, Jostein Nordengen, Kristian Schlanbusch(VikingBad), Magne Herlofsen(VikingBad), Caroline Berg

Sammendrag

Morgan oppsummerte krav til et Bachelorprosjekt på Multimedieteknologi og design-linjen hos UiA og oppdragsgiver sitt ansvar i forhold studenten og produkt.

VikingBad presenterte kort hvordan de så for seg et ønsket produkt kunne være for dem som bedrift.

VikingBad AS ønsker å stille som oppdragsgiver for prosjektet og får oversendt "Standardavtale om utføring av studentoppgaver Tekreal" fra Caroline.

I samråd med veiledere fra UiA og representantene fra VikingBad AS ble de ytre rammene for prosjektet formet.(Se punkt 2)

Caroline jobber videre med forprosjektet i slutten av januar. Sender over rapport så snart den er levert til veileder og representant for VikingBad AS.

Agenda

1.Bestemme oppdragsgiver for Bachelorprosjektet

VikingBad AS har sagt seg villig til å stille som oppdragsgiver i prosjekte med Kristian Schlanbush og Magne Herlofsen som representanter. De får tilsendt "Standardavtale om utføring av studentoppgaver Tekreal" før den sendes til UiA for signering.

2.Definisjon av viktige punkter for prosjektet

Hovedmål:

Det skal være en app som lastes ned på mobilen som benytter seg av mobilens innebygde kamerafunksjon og AR-teknologi til navigering og design. Applikasjonen skal gjennom realistiske 3D-modeller og interaktivitet - engasjere og inspirere brukeren til å lage et realistisk og navigasjonsbasert baderomdesign, basert på VikingBad AS sine produkter.

Prosjektet har 2 delmål som kan jobbes med parallelt eller vektlegges etter ønske:

1. Bruke AR-teknologi til å måle opp et rom, finne hjørnet og kanter og lage en 3D modell av rommet uten innredning.
2. Plassere 3D modeller på et plan utifra et sortiment og en meny. Modellene skal kunne legges til, flyttes på, roteres, festes og fjernes fra planet. Modellen skal beholde plasseringen på planen samtidig som kamera navigeres rundt på planet.

3. Plan for videre arbeid

Forprosjektet beregnes og bli ferdigstilt innen utgangen av januar. Innen den tid skal det tas valg av programvare, samt platform.

Prosjektet skal brytes ned i underpunkter med potensielle løsningsmetoder, og prioriteringsliste, samt Gant-diagram legges ved i forprosjektsrapporten.

Nytt møte gjennomføres etter at forprosjektet er levert inn. Endelig frist på forprosjektet er 25.februar 2021.

Møtereferat - Avtalevurdering og oppgaveendring

Møtested: Zoom. Tidspunkt: 11.feb 2021 kl.13.30-14.00

Tilstede: Morgan Konnestad, Jostein Nordengen, Caroline Berg

Sammendrag

Det ble besluttet at avtalen med VikingBad AS om å være oppdragsgiver for oppgaven ikke blir signert. UiA kommer til å stå som oppdragsgiver under gjennomføring av oppgaven.

Oppgaven blir, etter ønske fra Caroline Berg, omdefinert til å innebære mer forskning enn tidligere. Testing av bruk av innebygde sensorer i mobilene til navigasjon og AR-konsept på web kompatibelt for brukt på mobil.

Agenda

1.Diskutere avtale om "Overdragelse av immaterielle rettigheter" fra VikingBad AS og bedriften som oppdragsgiver.

VikingBad AS har sendt over en avtale om "Overdragelse av immaterielle rettigheter" som skal erstatte "Standardavtalen for studentoppgaver" laget av UiA.

Denne avtalen skal regulere de immaterielle rettighetene til oppgaven og oppgaveresultatene. VikingBad AS ønsker å fravike Standardavtalen og få enerett til å utnytte oppgaveresultatene slik de ønsker, inkludert å søke patent- eller designbeskyttelse for materialet.

Studenten, Caroline Berg, ønsker å utvikle et konsept som kan videreføres og distribueres etter prosjektets slutt og ønsker derfor ikke å overdra rettighetene til dette. Studenten signerer derfor ikke avtalen og gir bedriften beskjed om dette.

I tillegg ønsker studenten å endre på prosjektets målsetning, noe som også gjør at prosjektet ikke lenger er av en karakter hvor det er naturlig at VikingBad AS stiller som oppdragsgiver til oppgaven. Bachelorprosjektets oppdragsgiver blir flyttet internt og UiA står nå som oppdragsgiver.

2.Definisjon og målsetning for oppgaven

Etter ønske fra studenten blir hovedfokuset til oppgaven følgende:

- Utforske muligheten for å implementere mobilens innebygde sensorer til navigering og analysering.
- Testing av bruk av innebygde sensorer i mobilene til navigasjon og AR-konsept på web kompatibelt for brukt på mobil.
- Lage et interaktivt XR konsept som er kompatibelt med et flertall av mobileenheter.
- Dette innebærer blant annet:

1. Bruke AR-teknologi til å måle opp et rom, finne hjørnet og kanter og lage en 3D modell av rommet uten innredning.
2. plassere objekter i rommet som skal beholde den relative plasseringen etter navigasjon og rotering av enheten.

Caroline har fortsatt frihet til å definere målene nærmere og rangere disse.

3.Plan for videre arbeid

Definere prosjektets mål og legge en tidsplan for delmålene basert på en prosentvis andel av den totale tiden til rådighet.

Nytt møte avtales med veiledere ved behov ellers etter levering av forprosjektet 25.februar.

Møtereferat - Tilbakemelding på Forprosjektrapport

Møtested: Zoom. Tidspunkt: 09.mars 2021 kl.13.00-13.30

Tilstede: Morgan Konnestad, Jostein Nordengen, Caroline Berg

Sammendrag

Det ble snakket kort om forprosjektrapporten og alt var ok for begge parter. Plan for arbeid videre i prosjektet følges.

D Timeliste

Dato	Timer	Kommentar	Linker	Dato	Timer	Kommentar
3.1.2020	8	Forprosjekt - AR muligheter og inspirasjon	ViewAR		1.9.2021	4
4.1.2021	7	Research openSource AR, image tracking			6.9.2021	Heroku, GitHub, opprette enkel nettside for testing
5.1.2021	7	Research openSource AR, image tracking			7.9.2021	Lese sensordata på nettsiden, implementere Three.js, opprette scene med cube-objekt og helperGrid
6.1.2021	8	research unity, edge detection			8.9.2021	Navigasjon med gyroskop, deviceorientationControls, implementere video-element og hente inn baks-kamera.
7.1.2021	8	research edge detection			9.9.2021	2 Navigasjon med Accelerometer Integrasjon(Vel og Distance), kalibrering av accelerometer
8.1.2021	1	Møte med veiledere for godkjenning av prosjektet	Møtereferat 08_01		16.9.2021	5 kalibrering av accelerometer, Implementere retning av akselerasjonen, printe data for kontroll i CSV-fil
9.1.2021	7	Test av Unity og AR-plugins			17.9.2021	6 kalibrering av accelerometer, Implementere retning av akselerasjonen, printe data for kontroll i CSV-fil
12.1.2021	8	Test av Unity og AR-plugins			20.9.2021	7 kalibrering av accelerometer, Implementere retning av akselerasjonen, printe data for kontroll i CSV-fil
14.1.2021	1	Møte med VikingBad og veiledere UiA	Møtereferat 14_01		26.10.2021	5 Arbeid med rapport
15.1.2021	3	skriving av møtereferat og research til forprosjektet			27.10.2021	5 Arbeid med rapport
16.1.2021	2	Definisjon av deloppgaver			28.10.2021	5 Arbeid med rapport
18.1.2021		Database-design, flowchart, detaljert oversikt over oppgave			29.10.2021	5 Arbeid med rapport
21.01.2021	4	Research Database			30.10.2021	5 Arbeid med rapport
25.01.2021	5	Research Programvare			31.10.2021	5 Arbeid med rapport
29.01.2021	8	OpenSource Research			1.11.2021	5 Arbeid med rapport
4.2.2021		Forprosjekt, rapport og testing av modul for deviceMotion			2.11.2021	6 Arbeid med rapport
5.02.2021		deviceMotion, openSource, Aframe, Three.js og research på Isensorer for unity og unreal			3.11.2021	7 kalibrering av accelerometer, Implementere retning av akselerasjonen, printe data for kontroll i CSV-fil
6.02.2021		Implementering av DeviceMotion, DeviceOrientation og Three.js i test-moduler			4.11.2021	8 kalibrering av accelerometer, Implementere retning av akselerasjonen, printe data for kontroll i CSV-fil
7.02.2021		Implementering av DeviceMotion, DeviceOrientation og Three.js i test-moduler			5.11.2021	9 Arbeid med rapport
8.02.2021		Implementering av DeviceMotion, DeviceOrientation og Three.js i test-moduler			6.11.2021	10 Arbeid med rapport
11.02.2021		møte med veiledere, mail til VB og oppgavedefinisjon	Møtereferat 11_02		7.11.2021	11 Arbeid med rapport
12.02.2021	1	Skriv matereferat fra 11.02			8.11.2021	12 Arbeid med rapport
15.02.2021	10	Akselerasjon med deviceMotion - testing			9.11.2021	13 Arbeid med rapport
16.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			10.11.2021	14 Arbeid med rapport
17.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			11.11.2021	15 Arbeid med rapport
18.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			12.11.2021	16 Arbeid med rapport
19.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			15.11.2021	17 Rapport og OpenCV, Canny Edge Detection
20.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			16.11.2021	18 Arbeid med rapport
21.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			17.11.2021	19 Arbeid med rapport
22.02.2021		Forprosjektrapport - Research, planlegging og definering av prosjektet			18.11.2021	20 Arbeid med rapport
23.02.2021	8	Forprosjektrapport - Ferdigstillelse			19.11.2021	21 Arbeid med rapport
09.03.2021	1	Veiledningsmøte	Møtereferat 09_03		20.11.2021	22 Arbeid med rapport
					21.11.2021	23 Arbeid med rapport
					22.11.2021	24 Arbeid med rapport
					23.11.2021	25 Arbeid med rapport
					25.11.2021	26 Arbeid med rapport
					26.11.2021	27 Arbeid med rapport
					27.11.2021	28 Akselerasjonssteller
					28.11.2021	29 Harris corner, lagring av hjørner
					29.11.2021	30 Harris corner, lagring av hjørner
					30.11.2021	31 Harris corner, lagring av hjørner
					1.12.2021	32 Harris corner, lagring av hjørner
					2.12.2021	33 Arbeid med rapport
					3.12.2021	34 Harris corner, lagring av hjørner
					4.12.2021	35 Arbeid med rapport
					5.12.2021	36 Arbeid med rapport
					6.12.2021	37 Arbeid med rapport
					7.12.2021	38 Arbeid med rapport
					8.12.2021	39 Arbeid med rapport
					9.12.2021	40 Harris corner, lagring av hjørner
					10.12.2021	41 Arbeid med rapport, test med Akselerasjon
					11.12.2021	42 Dybdeutregning
					12.12.2021	43 Dybdeutregning
					13.12.2021	44 Arbeid med rapport
					14.12.2021	45 Arbeid med rapport
					15.12.2021	46 Arbeid med rapport
					16.12.2021	Innlevering
Timer vår 21:	204					
Timer Høst 21	357					
TOTALT:	561					

E Forprosjektrapport

Forprosjektrapport - Bacheloroppgave

Multimedieteknologi og -design
Vårsemesteret 2021

Bacheloroppgave: HPR/DM-016

Tittel: Bruk av mobiltelefon i AR

Utført av: Caroline Berg

Utgavedato: 14. desember 2021



Universitetet i Agder, 2020
Fakultet for teknologi og realfag
Jon Lilletuns vei 9, 4879 GRIMSTAD
Tlf. 37 25 30 00

Forord

Denne rapporten inneholder introduksjon og vurderinger gjort opp mot Bachelorprosjektet i DAT303 på Universitet i Agder våren 2021. Forprosjektet tar for seg prosjektets målsetning og ser på alternative løsninger av hovedprosjektet. Alternativene vurderes og anbefalt løsning vurderes ut ifra hvilken grad ønsket resultat oppnås gjennom utføringen. Prosjektplan for gjennomføring av hovedprosjektet blir satt basert på uker avsatt til prosjektet og deloppgaver som er nødvendig å gjennomføre for å få til en tilfredstillende resultat.

Målsetningen er tatt opp med veilederne Morgan Konnestad og Jostein Nordengen ved Universitetet og blitt godkjent. Delmål og løsningsalternativer er opparbeidet av studenten.

Nøkkelord: AR, 3D, navigasjon, applikasjon, prosjektplan, løsningsalternativer

Innhold

Forord	ii
1 Innledning	1
1.1 Bakgrunn	1
1.2 Prosjektets organisering	2
2 Prosjektets målsetning og rammebetingelser	3
2.1 Målsetning	3
2.1.1 Spesifikasjoner	3
2.2 Avgrensninger	3
2.3 Forutsetninger	3
3 Løsningsalternativene	4
3.1 Faktorene	4
3.2 Alternativ 1	6
3.3 Alternativ 2	6
3.4 Alternativ 3	6
3.5 Vurdering av løsningsalternativene	6
3.6 Anbefalt løsning	7
4 Plan for videre arbeidet	8
Referanser	9
A Prosjektplan	1

1 Innledning

Bacheloroppgaven går ut på å anvende aktuell kunnskap og teknologi tilegnet i studiet til å analysere og løse problemer eller behov for industri og offentlig sektor. Tema og problemstilling for prosjektet er valgfritt og skal godkjennes av veilederne. For å få et innblikk i industriens behov innenfor studieretningen - Multimedideteknologi- og design, er det ønsket at oppgaven omfatter temaer innen fagområdet. Det vil bli jobbet med og sett på temaer som 3D, bildebehandling, objektbasert programmering og webpublisering.

1.1 Bakgrunn

Under høstsemesteret 2020 ble det gjennomført en praksisperiode hos VikingBad AS. Dette var i forbindelse med faget IKT200 hvor studentene skal få prøvd seg ut i arbeidslivet gjennom å søke på en relevant stilling og gjennomføre praksistimer. I denne forbindelse fikk jeg tatt del i et prosjekt med å animere 3D modeller som skulle støtte opp om installasjonsmanualene til bedriftens moduler. Under perioden satte jeg meg mer inn i hvordan bedriften benyttet seg av 3D modellene ut mot sine kunder.

VikingBad har på sine nettsider en interaktiv modul hvor brukeren kan velge mellom stiler og stilpakker, tegne opp baderommet i 2D og 3D og få dette oversendt. I følge nettsiden til bedriften [1] er hensikten med denne modulen “gjøre det lekende lett” for brukerne å finne sin stil på baderommet.

Etter å ha testet ut tegneprogrammet og modulen føltes det litt tungvint for en som ikke har tegnet mange bad i sitt liv, og jeg satt også igjen med et ønske om å se resultatet i et mer levende format.

Dette var bakgrunnen for at jeg begynte å se på hva som kan gjøre applikasjoner mer interaktive med det nyeste innen dagens teknologi. Interaktivitet skal inviterer brukerne til å benytte flere av sine sanser under bruk av applikasjonen. De siste årene har det nyeste på applikasjon-fronten vært å skape en kobling mellom den virtuelle verdenen og virkeligheten. Derfor vil det være aktuelt å utforske muligheter og verktøy som benyttes innenfor VR, AR og generelt XR for å gjøre det interaktivt og tilgjengelig og brukbart i applikasjoner som skal distribueres til et variert spekter av brukere.

Ved videre undersøkelse finner vi at flere aktører har tatt i bruk 3D-løsninger for å designe både kjøkken og baderom. Compusoft[2] tilbyr en 3D planner for sine kunder både for kjøkken og baderom, mens IKEA[3] gir kundene tilgang til en Home Planner. Begge løsningene gir tilgang til å designe et rom nøyaktig og se produktene i et realistisk virtuelt rom.

Det er et slikt produkt prosjektet har som mål å utvikle og gjøre til en enda mer interaktiv for brukeren, samt gi et mer realistisk preg.

1.2 Prosjektets organisering

Prosjektet blir gjennomført med veiledning fra UiA ved Morgan Konnestad og Jostein Nordengen. Oppdragsgiver er UiA.

Det vil bli gjennomført jevnlige møter med samtlige veiledere da dette er et omfattende prosjekt som krever mange små og store avgjørelser. Støtte til eventuelle kostnader dekkes av oppdragsgiver etter avtale.

Under prosjektet blir det testet ulike metoder og programvarer. Kode og biblioteker blir samlet på et privat repo på GitHub for å kunne ha lett tilgang til koden. Under skriving av script eller kode blir Visual Studio Code benyttet da det allerede er etablert en arbeidsflyt gjennom tidligere bruk av dette programmet.

Ulike forslag til løsninger blir diskutert fram med veiledere, men vil bli testet på egenhånd ved å lage små, enkle prototyper med tilsvarende ønskede funksjoner.

2 Prosjektets målsetning og rammebetingelser

Prosjektets målsetning ble satt i samråd med veiledere og har sammenheng med eksisterende teknologi og teknologiske muligheter.

2.1 Målsetning

Ta i bruk en enhet med kamera for å filme et avgrenset området som blir grensene for et virtuelt rom som en kan plassere ut 3D objekter i. Navigasjon i det virtuelle rommet skal baseres på bevegelsen brukeren gjør med enheten.

Google Developers refererer til et liknende konsept utviklet av Sotheby's International Reality i en case study "Sotheby's International Realty: A Case Study" [4]. Applikasjonen [4] benytter seg av AR Core og skal tilby en mulighet til å designe og teste møbler i rommet før kjøp.

Se kapittel 4 for mer detaljerte delmål i dette prosjektet.

2.1.1 Spesifikasjoner

For at konseptet skal kunne være aktuelt for et marked må det settes noen krav til det endelige produktet.

Å sette krav om at forbrukerne må gå til innkjøp av spesifikke briller eller andre produkter for å ta i bruk applikasjonen kan fungere for bedrifter som driver med salg av dette, men ikke for bedrifter som selger møbler, interiør eller lignende. I dette prosjektet ønsker vi å fokusere på enheter som de aller fleste allerede har til rådighet - en mobiltelefon. Derfor vil applikasjonen kjøres på en nettleser og ved hjelp av mobilens innebygde sensorer.

2.2 Avgrensninger

- Applikasjonen testes på Android operativsystem
- Applikasjonen og navigering testes med ett 3D objekt
- 3D objektet skal kunne flyttes og roteres langs aksene

2.3 Forutsetninger

- Enheten må ha kamera med tilstrekkelig oppløsning
- Enheten må ha gyroskop- og akselerasjons-sensor
- Enheten må ha tilgang til en nettleser (Chrome eller Safari) og internett
- Testområdet må være godt belyst
- Testområdet må være fri for uønskede objekter

3 Løsningsalternativene

For oppgaven er det satt sammen tre forskjellige alternativer som kan sees i tabell 1. Faktorene er vektet forskjellig, noe som påvirker valg av endelig alternativ. Nærmere forklaring av faktorene er forklart videre i kapitelet.

Tabell 1: Tabell over alternativ 1-3

Faktor/Alternativ	Alternativ 1	Alternativ 2	Alternativ 3
Unreal/Unity	Ja	Nei	Nei
ARCore	Ja	Ja	Nei
Andre Plugins	Ja	Nei	Ja
Open Source	Ja	Ja	Ja
Kostnad	*	*	*
Platform	lokal	web	web
Krav til enhet	Ja	Ja	Nei

* Kostnadene på alle alternativene er så lave at de ikke tas med i vurderingen som en avgjørende faktor.

3.1 Faktorene

Unreal/Unity og ARCore: Unreal Engine[5] og Unity[6] er utviklingsplattformer for spill og applikasjoner, mens ARCore er Google sin egen plattform for utvikling av AR-applikasjon [7]. Gjennom disse plattformene har brukerne tilgang til mengder av plug-ins og funksjoner som bidrar til en effektivt sammenfatning av kode og funksjoner. Unreal Engine bruker ARCore og ARKit [8] og Unity bruker ARKit, ARCore, Magic Leap, og HoloLens [8]. Selv om slike verktøy byr på mange muligheter har de også begrensinger i form av type enhet de kan brukes på og måten de brukes på. På f.eks ARCore sine nettsider finnes det en liste over hvilke typer enheter som er kompatible [9], det er mange enheter, men de aller fleste må være av nyeste type. Ved bruk av ARCore i applikasjoner må enheten også ha ARCore installert lokalt.

Begrensningene som bruken av ARCore/ARKit gir på modifisering av koden, nedlastning av tilleggsapp, samt kompatibilitetskrav til enhetene trekker mye ned i totalvurderingen.

Andre Plugins: Å ta i bruk plugins eller tilleggspakker med ferdig kode kan spare tid i et prosjekt med mange funksjonaliteter. Unity og Unreal har både innebygde pakker som kan benyttes og mulighet for å legge til andre pakker. Det ligger utallige ferdige moduler og pakker som kan implementeres tilgjengelig også for OpenSource programmering. Disse er opprettet av profesjonelle- og hobby-programmtere over hele verden.

Å kunne ta i bruk ferdig kode er et pluss i forhold til prosjektet, men alle moduler som inkluderes må testet før bruk for å sikre riktig og god funksjonalitet.

Open Source: Open Source eller “Åpen kilde” er kode som ligger åpent for alle. Rettighetshaveren gir brukerne til å implementere, endre og publisere koden uten begrensninger. Dette gir brukerne mulighet til å utvikle programmer videre uten å “Finne opp hjulet på nytt”, men innebærer også at det er begrenset hvor mye koden er testet og verifisert.

Utvikling med OpenSource kan medføre ekstra testing av kode, men er et pluss ettersom utviklingen og publisering av kode i fremtiden ikke begrenses av lisenser.

Kostander: Unreal Engine[5] kan benyttes gratis til utvikling av applikasjoner, mens Unity har et tilbud for studenter[10] som også er gratis for perioden.

Andre kostnader som kunne forekommel er for eksempel en platform for drift av applikasjonen på WEB. I denne forbindelse vil Heroku[11] bli benyttet. De tilbyr en “Free and Hobby” plan[12] som kan benyttes til blant annet “Proof of concepts”, noe som er perfekt for oppgaven.

Som forklart over er de eventuelle kostnadene så små at utelates fra den totale vurderingen av alternativene.

Platform: Med plattform vises det til om applikasjonen kjøres som en lokal app som lastet ned på enheten eller om den kjøres i en nettleser. Dette medfører ulike krav til enhetene. Hvis applikasjonen skal lastes ned må den være kompatibel med det enkelte operativsystemet. Mens i en webapplikasjon må brukeren få forespørsel om tilgang til enhetens sensorer og kamera.

En webapplikasjon kan letttere implementeres i bedrifters eksisterende nettside, mens en applikasjon må lastes ned manuelt av brukeren.

Krav til enhet: Hvilke krav som settes til brukernes enheter og operativsystemer har mye å si for hvor stor andel av et marked som kan benytte seg av en applikasjon. Det er derfor en viktig faktor i vurderingen av alternativene.

3.2 Alternativ 1

Dette alternativet inkluderer bruk av Unreal eller Unity med ARCore samt andre plugins og openSource kode. Kostnadene i utviklingsfasen er tilnærmet lik null og applikasjonen kjøres lokalt på enheten i form av en app. Dette alternativet setter krav til enhet siden Unreal/Unity kompilerer forkjellige apper til forkkjellig type enhet (Android/iOS), og krever også at enheten har de nødvendige spesifikasjonene for kamera og bevegelsesensorer.

3.3 Alternativ 2

I alternativ 2 blir ARCore implementert i en webleser. Open-source kode brukes og det stilles også her krav til enhetens kamera spesifikasjoner og bevegelsesensorer på grunn av AR Core implementering.

3.4 Alternativ 3

Her bygges applikasjonen som en web-løsning med open-source kode som blandt annet *Three.js*. Her stilles det minimale krav til enheten. ARCore funksjonalitet som navigasjon og tracking må erstattes med annen kode.

3.5 Vurdering av løsningsalternativene

Basert på vektlegning av de forskjellig faktorene kan hvert løsningsalternativ vurderes opp imot hverandre.

Alternativ 1 vil resultere i en applikasjon kompilert fra Unreal Engine, Unity eller ARCore. Den negative siden av dette er at det krever mye vedlikehold og tilpasninger for å ha en egen app til hvert operativsystem. ARCore er heller ikke kompatibel på alle enheter. Positive sider ved en slik løsning er at det er veldig gode biblioteker og plugins for AR som er godt testet ut og som en vet fungerer veldig bra.

Alternativ 2 er en web-basert løsning med ARCore som er kompatibel med operativsystemenes standard nettleser, i tillegg til at ARCore er installert på enheten. Det som trekker denne løsningen ned er begrensningen med ARCore hvor ikke alle eldre enheter er kompatible. Positivt med denne løsningen er at den er web-basert og trenger mindre vedlikehold. ARCore er også et velkjent verktøy som bruker av mange aktører og er derfor et trygt alternativ.

Alternativ 3 er også basert på en web-løsning hvor *Three.js* benyttes for 3D fremvisning. Negativt med denne løsningen er at de fleste funksjoner rundt navigasjon og 3D animering må skrives selv. Dette er tidkrevende men kan på den andre siden tilpasses slik en ønsker det. Andre positive siden med denne løsningen er at det ikke stilles samme krav til enhet

som det gjør ved bruk av ARCore. Koden som danner ferdig produkt er også egenkomponert og stiller ikke krav til lisens for et tredjeparts software.

3.6 Anbefalt løsning

Grunnet behovet for å å gjøre applikasjonen tilgjengelig for mobil, men samtidig et flertall av brukerne faller valget på alternativ 3. Det skal her lages en webapplikasjon med navigasjon som baseres på enhetens sensorer og kamera uten AR Core. Applikasjonen kan kjøres på Heroku[11] ved hjelp av Node.js[13], og bygges i Visual Studio Code[14] og GitHub med blant andre pakker som Three.js[15] og OpenCV[16].

4 Plan for videre arbeidet

For å få gjennomført prosjektet innen fristen 25.mai og sikre et arbeid som leverer opp mot målsetningen er det laget en prosjektplan. Prosjektet er delt inn i deloppgaver med tidsfrister. Se prosjektplanen i vedlegg A.

I hovedtrekk går planen ut på å dele prosjektet inn i mindre deler med egne mål. Etterhvert som delene er på plass eller er gjennomført dokumenteres resultatet i rapporten.

De tre hoveddelene er følgende:

- 3D rom og navigasjon i Three.js:

Her er det snakk om å få til en helt enkel “scene” i nettleseren med tilhørende kode, og platformer. Node.js, Heroku, Git-repo. Det skal også kunne leses informasjon fra enheten(mobilén) inn på nettsiden. Bevegelsene til enheten skal påvirke bevegelsene til kameraet i det virtuelle rommet i nettleseren.

- Kamera og dybde:

I denne delen skal kameraet implementeres på nettsiden. Object tracking skal testes for å finne dybden i rommet, og benyttes for å skalere objektet og skalere rommet.

- Rom og anker:

For at navigering og interaksjon skal føles realistisk ønsker vi å kunne ta utgangspunkt i veggene fra et virkelig rom og benytte tracking for å gi veggen på det virtuelle rommet i riktig rotasjon og størrelse.

- Test og tuning:

I denne fasen skal applikasjonen testes for å sjekke om den overholder kravene til prosjektet, samt gjøres små endringer i koden for å forbedre funksjonalitet, kodestruktur og lignende.

- Rapport:

Rapporten skrives underveis og er en dokumentasjon som skal sikre muligheten for etterprøving av resultatet. Resultatene dokumenteres underveis og etter endt prosjekt.

Referanser

- [1] VikingBad AS. Interaktivt baderom, 2021.
<https://www.vikingbad.no/tegneprogram/interaktive-baderom>.
- [2] Compusoft Group. Compusoft, 2020. <https://www.compusoftgroup.com/no/>.
- [3] IKEA. Home planner ikea, 2021.
<https://kitchenplanner.ikea.com/no/UI/Pages/VPUI.htm?ignoreDeviceDetection=true>.
- [4] Google Developers. Sotheby's international realty: A case study, 2021.
<https://developers.google.com/ar/use-cases/SIR.pdf>.
- [5] Epic Games. About unreal engine, 2021. <https://www.unrealengine.com/en-US/faq>.
- [6] Unity Technologies. Unity, 2021. <https://unity.com/>.
- [7] AR Core. Arcore overview, 2021. <https://developers.google.com/ar/discover>.
- [8] Unreal Engine. Augmented reality overview, 2021
<https://docs.unrealengine.com/en-US/SharingAndReleasing/XRDevelopment/AR/HandheldAR/AROverview/index.html>.
- [9] AR Core. Arcore supported devices, 2021.
<https://developers.google.com/ar/discover/supported-devices>.
- [10] Unity Technologies. Priser for unity, 2021. <https://store.unity.com/plans-individual>.
- [11] Heroku. Cloud application platform, 2021. <https://www.heroku.com/>.
- [12] Heroku. Priser for heroku, 2021. <https://www.heroku.com/pricing>.
- [13] OpenJS Foundation. Node.js, 2021. <https://nodejs.org/en/about/>.
- [14] Microsoft. Visual studio code, 2021. <https://code.visualstudio.com/>.
- [15] Three.js, 2021. <https://threejs.org/>.
- [16] OpenCV Team. Opencv, 2021. <https://opencv.org/>.

Appendiks

A Prosjektplan

