NTNU
Det skapende universitet

Intelligente Brukergrensesnitt i Smarte Hjem

Andreas Arning Flakstad

Master i datateknologi
Innlevert: juni 2015
Hovedveileder: Asbjørn Thomassen, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Sammendrag og konklusjoner

Dette prosjektet har hatt som mål å utforske ulike brukergrensesnitt i smarte hjem. Spesielt har fokuset vært på å utforske *intelligente brukergrensesnitt*; alternative interaksjonsmetoder som kan ha økt effektivitet og som oppleves som mer naturlige i bruk. Dette ble utfordret gjennom å spørre hva slags programvare man ønsker i smarte hjem. Basert på logisk argumentasjon og empiriske bevis fra to studier, viste jeg at brukere av smarte hjem ønsker å lære om hjemmets tilstand, og de ønsker å utøve styring over hjemmet. Samtidig må privatliv og personvern beskyttes. Med dette utgangspunktet utforsket jeg hvordan gester og tale kan benyttes for å gi kommandoer, og hvordan informasjonen om hjemmets tilstand best mulig kan presenteres.

Jeg har vist at maskinlæring kan benyttes for å gi enkle sensorer en mer detaljert forståelse av abstrakte gester, enn det som eksplisitt kan programmeres. På bakgrunn av teknologiske begrensninger og sikkerhetsmessige problemer, har jeg vist at det i et hjem-scenario er mer aktuelt med begrenset talegjenkjenning enn naturlig talegjenkjenning. Jeg har vist at begrenset talegjenkjenning og maskinlærte gester i kombinasjon er en aktuell, multimodal måte å styre hjemmet på. Til sist har jeg vist hvordan et tradisjonelt brukergrensesnitt kan forbedres ved å designe rundt presentasjon av kontekstinformasjonen, framfor brukerinteraksjon. En prototype ble utviklet, der fokusert var å så mye informasjon som mulig om hjemmet gjennom teknikker fra grafisk design. Regler for hvordan å vise den mest essensielle informasjonen ble utforsket og implementert i et grafisk brukergrensesnitt, som forandrer seg dynamisk basert på kontinuerlige datastrømmer fra det smarte hjemmet.

Abstract

The goal of this thesis has been to explore user interfaces with regards to smart homes. In particular, intelligent user interfaces has been of focus; alternate methods of interaction, which might prove more efficient, effective and natural. This was challenged by asking what sort of software is needed in smart homes. Based on logical argumentation and empirical evidence from two case studies, I showed that users of smart homes seek to learn about the state of the home, as well as having the ability to exercise control over the home. Simultaneously, these features must not create issues regarding privacy or security. With this starting point I explored how gestures and speech could be used to issue commands, and how information regarding the state of the home could be presented. I have shown that machine learning can be used to give simple sensors a more detailed understanding of abstract gestures, than what can be explicitly programmed. Based on technological and security-related issues I have shown that limited speech recognition is more appropriate in home scenarios, than natural speech recognition. I have shown that the combination of limited speech recognition and machine learned gestures, is a possible, multimodal approach of controlling the home. Finally, I have shown that a more traditional user interface can be improved by designing around the presentation of context information, rather than user interaction. A prototype was developed with a focus on presenting as much smart home data as possible to the user, through techniques from graphical design. Rules for deciding which information is the most essential to present was explored and implemented in a graphical user interface, which changes dynamically based on continuous data streams from the smart home.

Anerkjennelse

Jeg ønsker først å takke min veileder Asbjørn Thomassen. Med dette prosjektet har jeg fått svært frie tøyler til å utforske mine interesseområder. Samtidig har Asbjørn hjulpet meg med å strukturere oppgaven og minnet meg på å ikke glemme den akademiske tilnærmingen. Asbjørn bidro også med ideér, blant annet å utforske bruken av flere enn en enkelt sensor.

Videre må jeg takke Ivar Flakstad og May-Britt Størseth, som begge har hørt tålmodig på mine ideér og tatt seg tid til å lese og kommentere på prosjektet. Ivar kom også opp med ideén om å anvende gesteteknologi i helseindustrien, hvilket jeg nevner i kapitellet om videre arbeid.

Til sist vil jeg vise takknemlighet til *open source*-miljøet. Uten fri tilgang til programvare ville eksperimentene jeg har utført vært svært vanskelige å gjennomføre.

Figurer

1 Lineær hypotese skiller datapunktene i to klasser 12
2 Ordinær sigmoidfunksjon 13
3 Det optimale hyperplanet i svart og de to støttevektorene i grønt 15
4 Støttevektormaskinens kostnadsfunksjoner 15
5 Stort panel med knapper skaper forvirring. 26
6 Sensorstørrelse: APDS-9960, montert på Sparkfun-brettet. 30
7 Sensor, level shifter og Arduino. 32
8 Dannelsen av en datavektorer. 33
9 Fire sensorer. 38
10 Gest: brukeren sveiper hånd mot venstre 44
11 De 10 ulike gestene 45
12 Vokabular 48
13 Multimodal input visualisert 49
14 42 gester 51
15 Dimmeeffekt 52
16 Brukergrensesnitt med fokus på interaksjon 53
17 Smart hjem som grafisk design 54
18 Apparater skrudd av og på 55
19 Gang med dør låst og ulåst 56
20 Soverom med lyset slått på og av 56
21 Scenario 1 57
22 Scenario 2 57
23 Scenario 3 58
24 Scenario 8 58
25 Resultatsutvikling for et utvalg algoritmer 61

Tabeller

1 Utstyrsliste eksperiment 1 31
2 Klassifisering av 10 gester 59
3 Klassifisering av de samme 10 gestene fra eksperiment 1 59
4 Klassifisering av 42 gester 60

Kodeeksempler

1 Dataprosessering 34
2 Splitte datasettene 34
3 Bakgrunnstråder 36
4 Overføre data fra gester 36
5 Håndtere multimodal inputdata 37
6 Dimme lys 39
7 Datastruktur 40
8 Atom 40
9 Hovedkonteiner HTML 41
10 Kjøkken HTML 42
11 Regel 42

Innhold

1 Introduksjon **1**

2 Teori & Relatert arbeid **4**

2.1 Smarte hjem 4

2.2 Maskinlæring 9

2.3 Brukergrensesnitt 16

3 Metoder & Implementasjon **25**

3.1 Min subjektivitet 25

3.2 Hypoteser 25

3.3 Design av eksperimenter 28

3.4 Implementasjon 29

4 Eksperimenter & Resultater **43**

4.1 Evalueringsstrategier 43

4.2 Eksperimentsutførelse 43

4.3 Resultater 59

5 Diskusjon & Konklusjon **61**

5.1 Diskusjon 61

5.2 Videre arbeid 65

5.3 Konklusjon 67

Bibliografi **69**

A Ressurser **72**

B Akronymer **73**

1 Introduksjon

Bakgrunn & motivasjon

En vårdag noen år fram i tid våkner du av alarmen som går av. Sola skinner inn gjennom soveromsvinduet, der persiennene ble hevet automatisk for tredve minutter siden for å gi en mer naturlig oppvåkning. Du tusler ut på badet, der lyset kommer på idet du entrer rommet. Varmen i baderomsgulvet har vært skrudd av gjennom natten, men har stått på i en tilstrekkelig periode før alarmen gikk av for å tilby en behagelig temperatur. Du vrenger av deg nattskjorta og legger den i skittentøyskurven for hvitt tøy. En sensor registrer at denne kurven nå er blitt full og en hvit klesvask blir automatisk igangsatt. Mens du står og steller deg får du lyst på hjemmebakte rundstykker og kaffe til frokost. Ved å ytre noen ord og utføre noen gester mot en liten plate på baderomsveggen får du skrudd på både ovnen og kaffemaskina på kjøkkenet. Vel ute på kjøkkenet bruker du en touch-skjerm for å finne fram en oppskrift på rundstykker. Underveis i matlagingen blir det nødvendig å lese videre i oppskriften fra touch-skjermen, men hendene dine er nå tilgrisede med deig. Heldigvis håndterer også denne touch-skjermen enkle gester og du blar deg videre i oppskriften ved å sveipe hånda over enheten. Hjemmet holder en kontinuerlig oversikt over matbeholdningen og hva som må handles inn, basert på ønsker om hvilke varer du alltid vil ha tilgjengelig i hjemmet. Etter frokost forlater du hjemmet, på vei til jobb. Døra låses automatisk bak deg og støvsugerroboten setter i gang med å rengjøre huset mens du er borte. Du får så en notifikasjon på telefonen. En app til hjemmet viser en grafisk framstilling av hjemmet og du ser at ovnen fremdeles står på. Du skrur av ovnen via app-en og fortsetter reisen til jobben.

Dette kan bli et vanlig scenario i mange hjem. Teknologien utvikles videre og det er mange områder av hjemmet som kan forenkles og forbedres; automasjon, energibesparing, hjelp til eldre og funksjonshemmede og forbedret interaksjon.

Jeg vil nå definere noen ord og uttrykk for å fjerne ambiguitet og legge et solid grunnlag for den videre diskusjonen.

Smarte hjem brukes i denne oppgaven som et samlebegrep for boliger der man kan program-

mere og styre miljøet og apparatene. Smarte hjem tilbyr funksjonalitet som fjernstyring, energieffektivisering, økt komfort, økt sikkerhet, automasjon og generelt enklere bruk. Jeg har valgt å bruke "smarte hjem", i stedet for "smarte hus" for å påpeke at diskusjonen omtaler faste oppholdssteder generelt.

Interaksjon er kommunikasjonen mellom mennesket og datamaskinen, som i denne sammenhengen styrer hjemmet. Interaksjon omfatter både instruksene brukeren gir, men også tilbakemeldingen datamaskinen svarer med.

Naturlig interaksjon føles instinktiv, forståelig, er enkel i bruk og gir rimelige resultater.

Effektiv interaksjon er praktisk i utførelse og gir raske resultater.

Intelligente brukergrensesnitt. Brukergrensesnitt er programvare som gjør det mulig for en bruker å kommunisere med en datamaskin. De fleste kjenner dette som grafiske grensesnitt som tillater direkte manipulasjon gjennom mus, tastatur eller touch. Intelligente brukergrensesnitt (IUI) er en nyere generasjon av grensesnitt, med evner som å kunne tilpasse seg ulike brukere dynamisk, forstå kontekstinformasjon rundt interaksjonen og å tilby hel eller delvis hjelp til å oppnå brukerens mål. Noen intelligente brukergrensesnitt kan også håndtere flere inputkanaler samtidig, såkalt *multimodal* input. Idéen er at multimodal input kan fremme en mer naturlig og effektiv form for interaksjon, for eksempel ved å håndtere at en bruker benytter touch og tale samtidig.

Problemformulering

Nå som begrepene er definert kan jeg presentere oppgavens problemformulering.

Kan intelligente brukergrensesnitt benyttes for å tilby en mer naturlig og effektiv interaksjon i smarte hjem?

Mål

Hovedmålene for denne oppgaven er å:

- Benytte maskinlæring for å utvide egenskapene og bruksområdene til enkle sensorer.
- Utforske nytteverdien og mulighetene til å bruke gester og tale i smarte hjem.
- Utfordre nåværende grafiske brukergrensesnitt for smarte hjem, ved å lage et brukergrensesnitt drevet av kontekstinformasjon, framfor interaksjon.

Disposisjon

Kapittel to vil greie ut om bakgrunnsteori og relatert arbeid i fagområdene oppgaven har tilknytning til. I kapittel tre presenteres et utvalg hypoteser rundt brukergrensesnitt i smarte hjem, hvordan disse kan utforskes med eksperimenter og hvordan disse eksperimentene ble implementert. Kapittel fire beskriver utførelsen og resultatene fra disse eksperimentene. I kapittel fem oppsummeres oppgaven med diskusjon, forslag til videre arbeid og konklusjon.

Bemerkning

Det finnes norske ord og uttrykk for flere av domene denne oppgaven svinger innom, men i mange tilfeller er det vanligere, både i akademia og i industrien, å benytte de engelske uttrykkene. Jeg vil derfor unngå de norske uttrykkene der jeg anser det som vanlig å bruke de engelske. En liste med akronymer finnes i appendix B.

2 Teori & Relatert arbeid

2.1 Smarte hjem

I introduksjonen definerte jeg smarte hjem til å være boliger der miljøet og apparater kan programmeres og styres. Dette er ikke ny teknologi. Muligheten for å kontrollere hus og hytter på avstand har vært tilgjengelig for de mest interesserte i lang tid. Peine (2008) viser til at idéen om smarte hjem har vært beskrevet siden 80-tallet, men har i de senere år fått en ny interesse i industrien. Styring i kontorbygninger har eksistert siden 70-tallet, med muligheter for å kontrollere lys, varme, elektrisitet og adgangskontroll fra et sentralt sted. Det tok noen år før man innså at de samme egenskapene kunne være ønskelige i private hjem. Forskjellen mellom et vanlig hjem og et smart hjem, er altså denne muligheten til å styre flere aspekter ved hjemmet gjennom en sentral enhet.

På markedet finnes det nå flere løsninger for å automatisere deler av hjemmet eller som tilbyr brukergrensesnitt for å styre funksjonalitet som lys og varme. Disse løsningene bidrar med å ivareta sikkerhet og å begrense energiforbruk. Det tilbys gjerne informasjon om hjemmet via en app til smarttelefonen slik at brukeren kan ha oversikten selv når han er bortreist. Ved å installere moderne enheter med internettilknytning, som termostater, lys, låser, sikkerhetssystemer og garasjeporter, kan brukeren avlese informasjon og styre enhetene med smarttelefonen.

Denne neste generasjonen apparater med internettilkobling bringer spennende muligheter, men samtidig utfordringer. *Tingenes internett* spås en enorm vekst de neste årene. Se for deg følgende scenario: alle elektriske enheter og apparater, lys, dører, persienner og et stort antall sensorer for å måle temperatur, bevegelse og tilstedeværelse; alle koblet til internettet. Det vil være en utfordring å håndtere alle disse dataene på en ansvarlig måte. Det vil være en utfordring å tilby gode måter for å lære om og styre hjemmet.

Det finnes ikke et eget fagområde for smarte hjem. Teknologiene og teknikkene som kan brukes for å skape smarte hjem har tilknytning til en rekke fagområder, som bygningsautomasjon, robotikk, nettverk, sikkerhet, menneske-maskin-interaksjon (HCI), kunstig intelligens (AI) og ambient intelligens (AmI).

AmI er spesielt interessant i kontekst av smarte hjem fordi det omhandler miljøer som

er sensitive og responsive til mennesker. Det er en visjon på hvordan vi bruker teknologi i omgivelsene for å støtte opp under hverdagslige aktiviteter. Ettersom enhetene som utgjør denne teknologien stadig blir mindre, og stadig blir bedre på å kommunisere seg i mellom, vil teknologien forsvinne inn i omgivelsene. Det eneste synlige vil være brukergrensesnittene vi har designet for å aktivt interagere med teknologien. Dette er en videreføring av framtidssynet Weiser (1991) skriver om, der teknologien forsvinner inn i omgivelsene og hverdagslivet inntil det er umulig å skille de fra hverandre. AmI omhandler alle miljøer vi befinner oss i og omhandler dermed også hjemmet.

For å være til hjelp for oss må hjemmet forstå omgivelsenens tilstand og det vil være nødvendig å analysere sensorinformasjon. Jeg har allerede nevnt enkle sensorer, som temperatur og bevegelse, og i kombinasjon kan disse danne et godt bilde av hva som foregår i hjemmet. To langt mer datarike input-kanaler er lyd og bilde. Ettersom vi mennesker kommuniserer med tale er det kanskje også naturlig at hjemmet skal forstå tale? Talegjenkjenning er et vanskelig problem, men det er et fagområde det er gjort store framskritt i de seneste årene. Jeg vil greie ut om bakgrunnsteorien til talegjenkjenning i kapittel 2.3. Tilsvarende til tale kan man få en datamaskin til å forstå visuell data gjennom kameraer. Augusto and Nugent (2006) omtaler *datasyn* som svært nyttig for å gjenkjenne mønstre i menneskers oppførsel, eller for å oppdage når noe galt har skjedd, som at en eldre bruker har falt. Video gir potensialet for å realisere virkelig smarte systemer som kan motta kommandoer gjennom gester, forstå hjemmets tilstand og gjenkjenne brukernes aktiviteter. Video produserer store datamengder og dette gir en større teknisk utfordring med henhold til lagringsplass og datautvinning, enn bruk av enklere sensorer. Heldigvis er det utviklet mange gode teknikker for å analysere bilder og vi er i ferd med å gå inn i en tid der begrensning på lagringskapasitet er et ikke-problem. Et eksempelprosjekt som gjør bruk av video er *Placelab-prosjektet*. Intille et al. (2005) brukte ni infrarøde kameraer, ni fargekameraer og atten mikrofoner spredd omkring i en leilighet. Ved bruk av bildebehandlingsalgortimer kunne de velge hvilke av datastrømmene som best fanget beboerens oppførsel til enhver tid. Desverre er bruken av kameraer et problem i hjemmescenariet. Dersom man velger å lage smarte hjem som benytter seg av analysering av lyd og bilde må det også innses at problemer relatert til brukerens følelse av privatliv og beskyttelse av personvern må håndteres.

Smarte hjem ønsker å tilby funksjonalitet relatert til gevinst innen økonomi og komfort, som at lys og varme skrues av og på automatisk avhengig av brukerens tilstedeværelse. Men det kan argumenteres for at den viktigste funksjonaliteten i smarte hjem er å bygge opp under en uavhengig livsstil for eldre og funksjonshemmede. Å utvide tidsperspektivet for eldre og funksjonshemmedes uavhengige livsstil, er en av de mest studerte tilnærmingene til smarte hjem. En god grunn er at dette er den gruppen mennesker som sannsynligvis vil ha størst nytte av et smart hjem. Muligheten til å fortsette å leve uavhengig og selvstendig i sitt eget hjem, framfor å bli innlagt på en institusjon, anses som svært verdifull. Et smart hjem kan hjelpe til med funksjonalitet som å gi påminnelser om at medisin må tas og å oppdage dersom noe har gått galt og automatisk ta kontakt med hjelpetjenester eller familie. Den andre gode grunnen til at denne tilnærmingen studeres mye, er restriksjonene denne brukergruppen har i sitt levesett. Når de forskjellige aktivitetene brukerene utfører kan telles på noen få hender blir det store problemet om å få et datasystem til å forstå menneskelige aktiviteter mye enklere. Det blir satt restriksjoner på problemet som gjør at det faktisk kan løses. Å løse problemet med å forstå vanlige menneskers oppførsel er en svært vanskelig oppgave; vi gjør ofte flere aktiviteter på en gang, vi samarbeider med andre mennesker og vi kan tilsynelatende tilfeldig avslutte påbegynte aktiviteter.

De Silva et al. (2012) påpeker at for systemene som fokuserer på støtte av eldre og funksjonshemmede er det ekstra viktig å detektere mennesker, ettersom en av hovedfunksjonalitetene til et slikt system er å gjenkjenne hvilken tilstand brukeren er i. Dersom brukeren for eksempel har falt er det viktig å gjenkjenne objektet på bakken som et menneske og ikke som en livløs gjenstand. Ettersom å gjenkjenne mennesker er av høyeste prioritet benytter mange av prosjektene i denne kategorien videoovervåkning. Elliott et al. (2009) omtaler omtaler et alternativ til video ved å bruke en pendel sammen med refleksjons-sensorer for å følge pendelens posisjon i reell-tid. Resultatene indikerte at teknikken er tilstrekkelig sensitiv og kan brukes til å gjenkjenne forskjellene mellom en person som finner balansen og en som er i ferd med å falle. Dette er en måte for hjemmet å gjenkjenne og potensielt forhindre et fall før det skjer, uten videoovervåkning. Farella et al. (2010) skriver om kombinasjonen av sensorer fordelt i hjemomgivelsene og bærbare enheter, for å overvåke brukerens helse og aktivitet. Målet er å tilby overvåkning for å forbedre sikkerheten og livskvaliteten til

eldre mennesker som bor alene, uten å være påtrengende. Sensorene tillater overvåking av de typiske funksjonene i et avansert hjem: tilgangskontroll, gasslekkasjer, lys, lyd, åpne vinduer, fuktighet og temperatur. Interaksjonen mellom de utplasserte sensorene og de bærbare enhetene tillater innendørs sporing av menneskene og muligheten til å oppdage farlige hendelser. Sporingen realiseres ved å benytte signalstyrken mellom den bærbare enheten og sensorene i omgivelsene. Systemet kan også oppdage om uautoriserte personer befinner seg i hjemmet. Dersom en person oppdages ved en infrarød sensor og systemet ikke registrerer en bærbar enhet i det samme området kan en alarm aktiveres. Alarmen kan få brukerens bærbare enhet til å vibrere eller aktivere et web-kamera i nærheten for å tilby informasjon til slektninger eller andre. Systemet kan også detektere fall ved kombinasjonen av et akselerometer i den bærbare sensoren som kan si om personen ligger, og sporingen som kan si om personen befinner seg på soverommet eller ikke. Dersom et fall registreres kan systemet vibrere den bærbare enheten. Dersom brukeren ikke stopper alarmen går den videre og systemet tar kontakt med utenforstående og skrur på web-kamera for å tilby innsyn til hjemmet. Dette er også en tilnærming som unngår videoovervåkning.

Energisparing er et annet viktig tema innen smarte hjem. Energiforbruket kan minimeres blant annet ved å automatisk skru av lys og varme når det ikke trengs. Et smart hjem kan enten aktivt gå inn og skru ned eller av lys, varme og apparater når de ikke er i bruk, eller det kan overvåke energibruken og periodisk komme med forslag til forandring i brukerens holdning til bruk av elektrisitet. To store prosjekter som fokuserer på energieffektivitet er *MavHome*- og *Thinkhome*-prosjektene. Cook et al. (2006) skriver at MavHome har hatt som mål å skape et hjem som oppfører seg som en rasjonell agent og som forsøker å oppnå to mål samtidig: å maksimere beboernes komfort og å minimere kostandene for å operere hjemmet. For å nå disse målene må agenten ha evnen til å forutse bevegelsesmønstrene og bruken av elektriske enheter blant brukerene. Individuelle agenter kan ta seg av en del av problemet, men må koordinere deres handlinger for å oppnå det overordnede målet. Reinisch et al. (2011) påpeker at tidligere løsninger på smarte hjem ikke har klart å holde energinivået lavt og samtidig tilbudt god komfort for beboerne, og at Thinkhome kan være løsningen. Thinkhome benytter en stor kunnskapsbase for å ta vare på den nødvendige informasjonen for energieffektivitet og brukerkomfort, og anvender også rasjonelle agenter for å bygge et

intelligent system.

Jeg har hittil omtalt prosjekter som fokuserer på hjelp for eldre og energieffektivitet. La oss vende blikket mot hva brukere flest ønsker fra smarte hjem. Smarte hjem er et nytt produkt for det store markedet. Det er derfor uvisst om brukere vet hva de vil ha fra et smart hjem, eller om de vil ha et smart hjem i det hele tatt. I stedet for å dytte behov på brukeren kan det være bedre å støtte opp under brukerens faktiske behov. Dette kan hjelpe oss å bygge produkter som blir godt motatt og dermed kan vi framskynde innføringen av smarte og automatiserte hjem. En interessant, italiensk studie utfordret en gruppe på hva de ville spurt hjemmet sitt dersom det var intelligent. Bonino and Corno (2011) viste at folk flest har sterke følelser knyttet til hjemmet: det er et trygt og koselig sted, det er et sted til å stole på og følelsen av å returnere dit er positiv, å føle seg bra en del av hjemopplevelsen, atmosfæren er behagelig og tilpasset brukerens preferanser og folk kan gjøre hva de selv vil. Spesifikt ønsket brukerne i undersøkelsen tilgang til klokke-, kalender og værinformasjon, samt informasjon om energiforbruket i hjemmet og hvordan det kunne reduseres. De ønsket å kunne styre hjemmets underholdningssenter, regulere lys, temperatur og persienner, og stemmekontrollere hvitevarer. De ønsket også at hjemmet kunne gjøre tilgjengelig lesing av epost, bruk av telefon og ha evnen til å hjelpe med å huske ting og å søke opp informasjon. De ønsket automatisk oppdagelse av farer, som innbrudd, røyk-, varme- og gassutvikling. Hjemmet skulle overvåke seg selv og automatisk oppdage og reparere problemer. Til sist ønsket de hjelp til husholdningsoppgaver, håndtere matvarer og planlegge innkjøp.

Röcker et al. (2004) omtaler en annen studie av brukernes behov fra smarte hjem. Deltakerne hadde her et enda større fokus på at det er mennesket som skal ha kontroll over hjemmet. De var også enige i at funksjonalitet som energibesparing, brukergrensesnitt mot hvitevarer, husholdningsoppgaver og planlegging var viktig. Men foran alt dette plasserte disse deltakerne sikkerhet, beskyttelse av personvernet og fokuset på at et slikt system måtte tilføre ny verdi og ikke komme i veien for direkte kommunikasjon mellom mennesker.

Disse forskningsresultatene leder mot utvikling av programvare som gjør det enkelt å automatisere, å se hjemmets status og å gi hjemmet kommandoer. Det hele må gjøres med en høy grad av sikkerhet og med bevarelse av personvernet. Dette peker mot at programvare og data, i hvert fall delvis, må holdes lokalt. Deltakerne i studiene har vist sterke motsetninger

mot overvåking i sitt eget hjem. Selv dersom det kan garanteres at dataene fra kameraer og mikrofoner holdes lokalt, kan følelsen av at personvernet er utsatt være nok til at brukerene holder seg unna smarte hjem. Det gir intuitivt mening at de færreste brukere ønsker kameraer som filmer dem overalt i hjemmet. Selv dersom det kunne garanteres at informasjonen aldri forlot hjemmet, eller at den kun blir lagret i en kort tidsperiode, vil tilsynelatende konstant overvåking være noe mange vil sette et stort spørsmåltegn ved. Noen av oss har fremdeles kvaler mot en *Orwelliansk* tilværelse der storebror kan se oss[1].

Et virkelig smart hjem kan lære av å observere brukerne. Å lære brukernes oppførsel vil være essensielt for at systemet skal forbedre seg selv over tid, og for å tilby en individuelt tilpasset opplevelse. Forskjellige brukere vil ha forskjellige tilstander, preferanser og vaner, og disse bør tas med i beregningen for at systemet skal være verdifullt.

2.2 Maskinlæring

La oss begynne med en definisjon på maskinlæring fra Arthur Samuel (1959): "Maskinlæring gir datamaskiner evnen til å lære uten å bli eksplisitt programmert." Denne definisjonen er over 50 år gammel, men den fanger hva vi er ute etter; interessen i å få datamaskinene til å lære å gjøre nyttige ting uten å behøve å fortelle dem eksplisitt hvordan hver enkelt oppgave skal utføres. I situasjoner med økende kompleksitet blir det raskt vanskelig, og til slutt umulig, å eksplisitt programmere en algoritme som skal løse problemet. Maskinlæring kan anvendes i disse situasjonene og kan noen ganger finne løsninger på problemet.

Alle maskinlæringsproblemer kan konseptualiseres som å finne en funksjon som knytter input til output. Målet kan være å tilnærme en enkel matematisk funksjon, det kan være å spå aksjekursen basert på historisk data eller det kan være å gi sannsynligheten for et en epost er spam, basert på innholdet. Man tar erfaring, i form av empirisk data, og bruker en algoritme til å finne en funksjon som dekker denne kunnskapen best mulig.

Maskinlæringen som er aktuell i dette prosjektet kalles overvåket læring. Målet er å klassifisere nye data korrekt, basert på treningsgrunnlaget fra tidligere data. Læringen sies å være overvåket fordi vi bidrar med informasjon om hvilke klasser som hører til hvilke data

[1] *1984*, George Orwell (1949)

i treningseksemplene. Framgangsmåten er å mate maskinen med mange eksempler på denne koblingen mellom data og klasse, og håpe at maskinen finner en matematisk funksjon som tilnærmer denne sammenhengen godt.

I eksperimentene som skal utføres i dette prosjektet må dataene dannes manuelt. Dette betyr at vi vil få relativt få treningseksempler, og sannsynligvis vil antallet datapunkter i hvert treningseksempel være større enn antallet treningseksempler. Basert på disse karakteristikkene er det sannsynlig at enkle, *lineære modeller* vil gi de beste resultatene[2]. Mer avanserte klassifiseringsteknikker, som for eksempel nevrale nettverk, kan i teorien tilnærme enhver funksjon, men de trenger langt flere treningseksempler for å finne de sanne sammenhengene mellom data og klasser.

Framgangsmåten for å lære er altså å mate et treningssett av datapunkter til en valgt læringsalgoritme. Algoritmen danner seg en hypotese om hva slags modell som best beskriver dataene. Denne hypotesen kan så benyttes til å gjøre gjetninger på nye datapunkter. Hypotesen avhenger av egenskapene i datapunktene. I en lineær modell er hypotesen $h_\theta(x)$ en funksjon av dataene x i treningseksemplene, der hvert datapunkt blir vektet av θ-verdier:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{1}$$

La oss modellere x til å være en vektor med n datapunkter:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ .. \\ x_n \end{bmatrix} \in R^{n+1},$$

[2] http://scikit-learn.org/stable/tutorial/machine_learning_map/

Vi lar θ være en tilsvarende vektor:

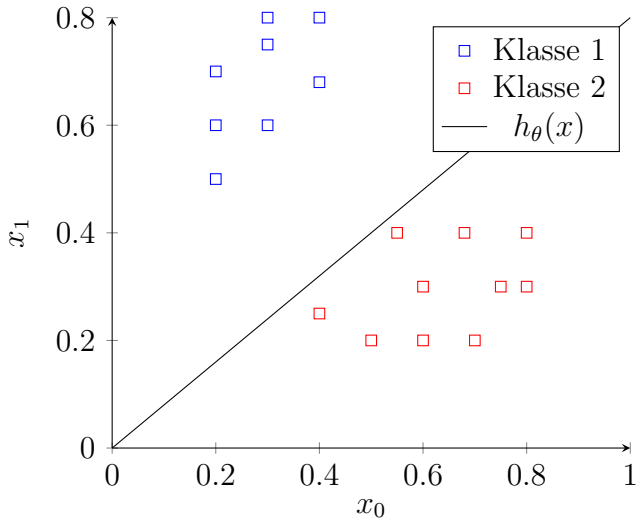
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ .. \\ \theta_n \end{bmatrix} \in R^{n+1}$$

Dersom vi nå tar den transponerte av θ og lar $x_0 = 1$, kan vi i stedet for (1) skrive hypotesen elegant og kompakt som indreproduktet av vektorene:

$$h_\theta(x) = \theta^T x \tag{2}$$

En vellykket hypotese vekter θ-verdiene slik at algoritmen gir de beste mulige resultatene. Søket etter den optimale vektingen tilsvarer det å finne hvilke datapunkter i treningseksemplene som er de viktigste for å angi hvilken klasse treningseksempelet tilhører. Så hvordan velger man disse θ-verdiene? Vi ønsker θ-verdier slik at hypotesen $h_\theta(x)$, er nære klassen y, for treningseksemplene (x, y). Treningseksemplene (x, y) representerer eksempelkoblinger mellom data og klasse. Dersom vi antar at hypotesen er en lineær funksjon og at treningseksemplene kun har to datapunkter, kan vi plotte hypotesen som en linje gjennom datapunktene. For hvert nye treningseksempel algoritmen prosesserer vil θ-verdiene justeres og linjen følger datapunktene som angir treningseksempelets klasse bedre. Etter at funksjonen er trent på en rekke treningseksempler av begge klassene, vil linjen forhåpentligvis danne et klart skille mellom datapunktene (se figur 1).

Separatoren i figur 1 er en linje i to dimensjoner. En separator i tre dimensjoner vil danne et plan. Å forestille seg en separator i mer enn tre dimensjoner er vanskelig. Heldigvis kan matematikken hjelpe da den ikke bryr seg om våre visuelle begrensninger og fungerer like godt i 128 dimensjoner som i tre. Vi har konkludert med at en lineær modell bør få sjansen til å skille treningseksemplene våre, men hvilken algoritme bør vi benytte? La oss utforske to av de mest brukte og robuste, lineære teknikkene: logistisk regresjon og støttevektormaskiner.



Figur 1: Lineær hypotese skiller datapunktene i to klasser

Logistisk regresjon

I logistisk regresjon modelleres sannsynlighetene som beskriver ulike utfall av en logistisk sigmoid-funksjon(3).

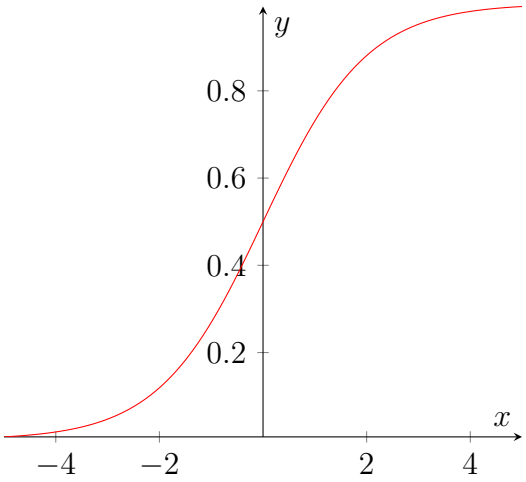
$$g(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Denne funksjonen tar en hvilken som helst input-verdi og gir en output-verdi i området $[0, 1]$. Likning 3 og figur 2 representerer den generelle *sigmoid*-funksjonen. Vi ser at ved større positive x-verdier gir funksjonen et resultat nære 1, mens for større negative verdier gir funksjonen et resultat nære 0. La oss si vi har to mulige klasser, $y \in \{0, 1\}$. Hvis $h_\theta(x) \geq 0.5$, gjetter vi at klassen $y = 1$. Hvis $h_\theta(x) < 0.5$, gjetter vi $y = 0$. Når vi nå kombinerer (2) og (3) får vi den logistiske hypotesen (4).

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{4}$$

Igjen representerer θ vektingen av egenskapene x i treningsdataene.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y) \tag{5}$$



Figur 2: Ordinær sigmoidfunksjon

(5) viser formelen for logistisk regresjon. Vi ser at $J(\theta)$ er et gjennomsnitt av en kostnadsfunksjon, definert av hypotesen til hvert treningseksempel og den tilhørende korrekte klassen.

$$cost(h_\theta(x^{(i)}), y) = \begin{cases} -log(h_\theta(x)), & \text{hvis } y = 1. \\ -log(1 - h_\theta(x), & \text{hvis } y = 0. \end{cases} \tag{6}$$

Kostnadsfunksjonen (6) er definert med delt forskrift slik at kostnaden er 0 dersom klassen er 1 og hypotesen er 1, men dersom klassen er 1 og hypotesen går mot 0, går kostnaden mot uendelig. Den delte forskriften gjør at det samme gjelder for det motsatte tilfellet, der kostnaden er lav dersom klassen er 0 og hypotesen gjetter 0, men øker mot uendelig dersom hypotesen går mot 1.

$$\theta_j \leftarrow \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta) \tag{7}$$

For å tilpasse θ-verdiene er strategien å minimere kostnadsfunksjonen (6). Dette gjøres ved å benytte en oppdateringsregel. Regelen oppdaterer egenskapsvektoren θ ved å trekke fra resultatet fra den partiellderiverte av kostnadsfunksjonen, dempet av en faktor α (7). Ettersom den deriverte i et gitt punkt kan sees på som brattheten til kurven i det punktet vil denne oppdateringen tilsvare å stadig ta mindre steg i den retningen som fører mot en lavere verdi. På en to-dimensjonell graf vil det si å følge plottet nedover mot et bunnpunkt, men algoritmen fungerer på samme vis i høyere dimensjoner. Denne oppdateringsregelen kalles

gradient descent og brukes i flere maskinlæringsalgoritmer for å finne minimumsverdier.

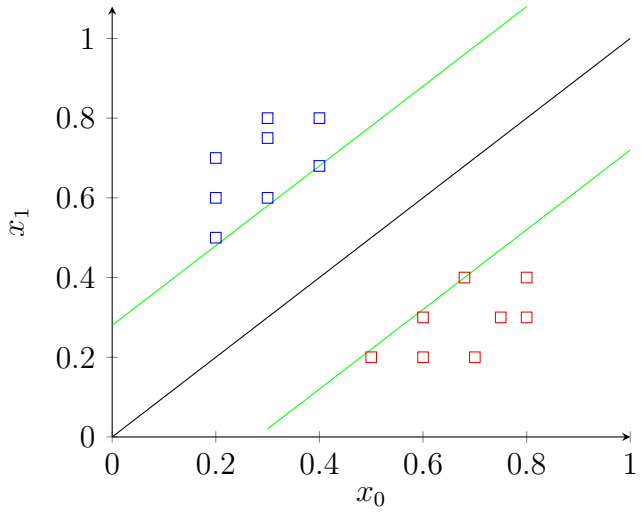
Etter at θ-verdiene er tilpasset av treningsdata kan modellen gjette hvilken klasse et nytt datapunkt tilhører ved å benytte hypotesen (4). For å lære å skille mellom mer enn to ulike klasser benyttes strategien "en-mot-resten". For hver klasse trenes det en egen hypotese som best mulig skiller mellom denne klassen og alle de andre. Når ny data kommer inn til det trente systemet velges den hypotesen som gir den høyeste sannsynligheten for en viss klassifisering og som dermed er mest sikker på å ha funnet det riktige svaret. I tillegg til å fortelle hvilken klasse dataeksempelet tilhører kan logistisk regresjon fortelle hvor sikker klassifiseringen er. Dette er en god egenskap som støttevektormaskiner mangler.

Støttevektormaskiner (SVM)

Støttevektormaskiner er en annen gruppe med maskinlæringsalgoritmer som kan brukes til å løse klassifiseringsproblemer. De er kjent for å være effektive i problemområder med mange dimensjoner og kan oppnå gode resultater selv når antallet datapunkter er høyere enn antallet treningseksempler. De bruker mindre plass i minnet enn andre algoritmer og kan tilpasses til å løse en rekke forskjellige problemer. To ulemper med SVM-er er at de ikke tilbyr direkte estimater på hvor sikker klassifiseringen er og at de, som logistisk regresjon, i utgangspunktet bare kan skille mellom to klasser.

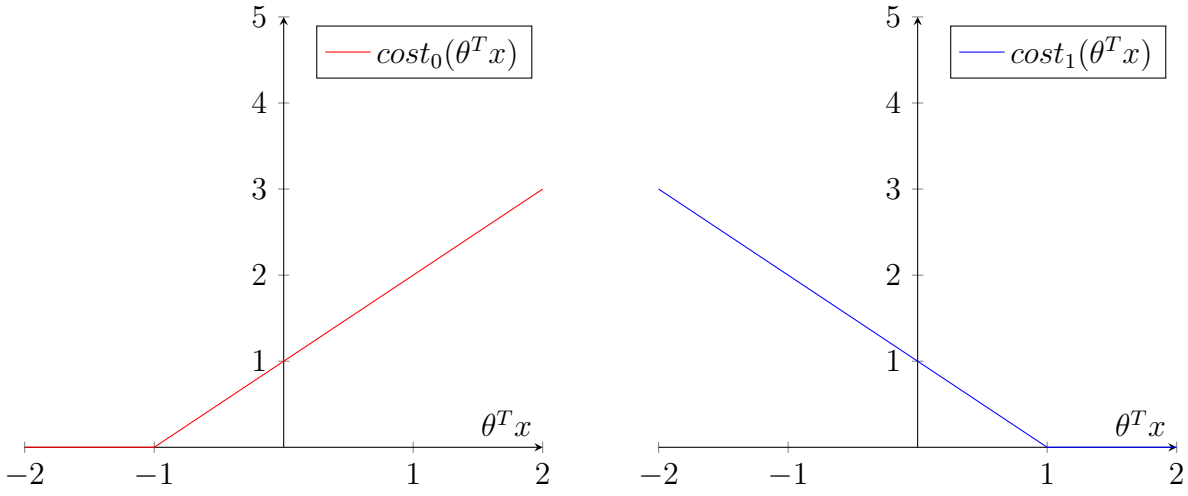
I figur 1 så vi en lineær hypotese som tydelig delte treningseksemplene i to klasser. Den separerende hypotesen ligger nærmere treningseksemplene for klasse 2. Dette er et resultat av at det er flere treningseksempler av denne typen. Dermed har den lineære metoden produsert en hypotese som ligger nærmere disse treningseksemplene. Vi kan også intuitivt forstå at det må være et uendelig antall forskjellige hypoteser som kan dele datapunktene, men at en hypotese som ligger midt mellom de to klassene av datapunkter vil være den mest robuste. Støttevektormaskiner benytter nettopp denne intuisjonen for å finne en optimal hypotese. Med støttevektormaskiner kalles separatoren et hyperplan som kan danne skiller i flerdimensjonale rom. En optimal deling oppnås av det hyperplanet som har den største avstanden til det nærmeste treningseksempelet hos hver klasse. Denne strategien om å finne den største marginen mellom klassene senker generelt klassifikatorens feilaktighet. Figur 3 viser et slikt

optimalt hyperplan som befinner seg der hvor marginene til hver klasse er maksimal og like stor. Treningen av støttevektormaskiner følger det samme mønsteret som logistisk regresjon,



Figur 3: Det optimale hyperplanet i svart og de to støttevektorene i grønt

men skiller seg på å ha en annen hypotese og kostnadsfunksjon. Hypotesen er det lineære indreproduktet vi så fra introduksjonen om klassifisering (2). Kostnadsfunksjonen er enklest forstått gjennom et plot. (4) viser de to kostnadsfunksjonene. Dersom klassen $y = 1$ ønsker



Figur 4: Støttevektormaskinens kostnadsfunksjoner

vi at hypotesen $\theta^T x \geq 1$. Dersom klassen $y = 0$ ønsker vi at hypotesen $\theta^T x \leq -1$

Treningen består dermed igjen av å minimere (8) med den samme oppdateringsregelen som for logistisk regresjon (7), men med en ekstra, justerbar parameter C som avgjør hvor

mye man ønsker å unngå å feilklassifiere hvert treningseksempel. Med en stor C-verdi vil optimaliseringen velge et hyperplan med mindre marginer, dersom dette hyperplanet gjør en bedre jobb på å få alle treningsdataene klassifisert riktig. En lav C-verdi lar optimaliseringen finne et hyperplan med større marginer, selv dersom dette hyperplanet feilklassifiserer flere treningseksempler.

$$J(\theta) = C \sum_{i=1}^{m} y^{(i)} cost_1(h_\theta(x^{(i)})) + (1 - y^{(i)}) cost_0(h_\theta(x^{(i)})) \tag{8}$$

Den typen SVM vi har presentert her er en såkalt lineær SVM eller en SVM uten kjerne. Ved å benytte et såkalt *kernel trick* kan SVM-er også modellere ikke-lineære funksjoner. Dette gjør SVM-er svært fleksible til å håndtere ulike data. I dette prosjekt er vi kun interessert i de lineære modellene. For å håndtere klassifisering av flere klasser benyttes gjerne den samme "en-mot-resten"-strategien som i logistisk regresjon. Denne ender altså opp med å trene n klassemodeller og modellen med det beste resultatet benyttes.

2.3 Brukergrensesnitt

Victor (2006) påpeker at programvare eksisterer for å hjelpe folk med tre ting: å lære, å skape eller å kommunisere.

- *Informasjonsprogramvare* hjelper folk med å lære. Vi bruker informasjonsprogramvare for å konstruere og manipulere en intern representasjon av informasjonen.

- *Manipulasjonsprogramvare* hjelper folk med å skape. Vi bruker manipulasjonsprogramvare for å skape og manipulere en virtuell modell, representert i datamaskinen. Denne typen programvare kan bli forstått som et virtuelt verktøy, som en malingskost eller skrivemaskin, som benyttes som et grensesnitt mellom skaperen og sluttproduktet.

- *Kommunikasjonsprogramvare* hjelper oss med å kommunisere. Vi bruker kommunikasjonsprogramvare for å skape og manipulere en intern modell som er delt, forstått og synkronisert mellom flere mennesker. Kommunikasjon kan sees på som å lage et svar

på tillært informasjon. Den eksterne modellen en bruker kommuniserer, er den interne modellen brukeren har lært.

Brukere i smarte hjem trenger programvare til å gi kommandoer til hjemmet, som å skru på lyset i et rom eller låse utgangsdøra. De trenger også programvare for å lære om hjemmets tilstand, som å se temperatur, status på oppvaskmaskina eller om det mangler matvarer i kjøleskapet. For å lære om hjemmet bør det designes informasjonsprogramvare. For å gi kommandoer til hjemmet passer kommunikasjonsprogramvare best. Hvordan kan relevant informasjon om hjemmet kan presenteres for brukeren? Hvordan kan man effektivt og naturlig kan gi kommandoer til hjemmet?

I introduksjonen definerte jeg IUI til å være grensesnitt mellom mennesker og datamaskiner med et mål om å være mer effektive og mer naturlige i bruk, enn tradisjonelle grensesnitt. Kaufmann (1998) påpeker at dette oppnås gjennom å representere, resonnere og handle på modeller av brukeren, domenet, oppgaven, diskusjonen og mediumet. IUI er en del av HCI, ergonomikk, kognitiv vitenskap og AI. Målet til HCI er å gjøre datamaskiner mer hjelpsomme og enklere å bruke. Lieberman (2009) skriver om hvordan HCI og AI er relaterte og hvordan de bør samarbeide og kan lære av hverandre for å løse målene våre. Det må inses at det er *trade-offs* mellom systemets pålitelighet og grad av konsekvens, og systemets evne til å forstå hva brukeren ønsker og å tilby hjelp. Lieberman påpeker videre at avanserte brukergrensesnitt muligens trenger *tutorials* for å gjøre brukerne komfortable med systemet. Brukerens behov forandrer seg stadig så å ta utgangspunkt i et dynamisk system gir fleksibilitet til å håndtere framtidige forandringer. HCI har investert mye i forskning på modeller som *GOMS* eller *key-stroke*, for å modellere sammenhengene mellom oppgaver, og operasjonene som trengs for å fullføre dem i et gitt grensesnitt. Disse modellene hjelper med å evaluere effektiviteten i brukergrensesnittene. Å introdusere intelligens i systemene kan la brukere unngå utførelsen av en rekke operasjoner for å nå et mål. Brukeren kan i stedet angi et overordnet mål og det er så opp til systemet å finne ut hva som må gjøres for å oppnå målet.

Smarttelefoner og andre touch-skjermer

Touchskjermer i form av smarttelefoner og nettbrett er nå vidspredte og populære. Det er en interaksjonsform folk bør være komfortable med å bruke i det smarte hjemmet. At nesten alle har smarttelefoner med seg til enhver tid gjør også at muligheten til informasjon om og styring av hjemmet alltid er tilgjengelig.

Koskela and Vaananen-Vainio-Mattila (2004) undersøkte forskjellige brukergrensesnitt for å styre smarte hjem. De evaluerte bruken av en pc, en tv med fjernkontroll og en mobiltelefon. Resultatene viste at en pc fungerte best som en sentral enhet for å kontrollere aktiviteter som kan planlegges eller bestemmes på forhånd. Dette gjaldt å sette opp automatiseringer, slik som at gardinene skal trekkes for og at lysene skrus på til et visst tidspunkt. Mobiltelefonen ble funnet til å være det beste alternativet for direkte kontroll. Denne studien ble gjort i 2004 og resultatene kan derfor antas å være noe daterte, men resultatet om at en bærbar, mobil enhet var det beste for å direkte kontrollere omgivelsene er nyttig. Dagens smarttelefoner er kapable til langt mer enn mobiltelefonen fra 2004. Det virker derfor rimelig å konkludere med at bærbare touch-skjermer er et godt alternativ for å tilby interaksjon med hjemmet.

Omgivelsesskjermer

I kapitellet om smarte hjem nevnte jeg AmI og teknologi overalt i miljøet. En teknologi det forskes på i AmI er omgivende skjermer. Informasjonskilder, andre mennesker og omgivelsene skal kunne sees og interageres med når det er behov for det. Å ha informasjonskilder tilgjengelig forskjellige steder i hjemmeomgivelsene kan gi nye muligheter for samarbeid mellom mennesker i aktiviteter som omhandler både lek og arbeid.

Forskjellige interaksjonssoner beskrives av Streitz et al. (2005). De omtaler interaksjonssonene omgivende, notifikasjon og interaksjon. Avhengig av hvilken sone en bruker befinner seg i vil omgivelsesskjermen regulere hvordan informasjonen og interaksjonsmetoden presenteres. *Proxemics* foreslås av Greenberg et al. (2004) som en tilnærming for å gi enheter en mer naturlig oppførsel. De stipulerer at brukere naturlig forventer at når enhetene deres nærmer seg andre enheter eller gjenstander i omgivelsene, øker tilkoblingen og interaksjonsmulighetene forbedres. Bruken av soner gir mulighet til å tilby forskjellige visninger og

interaksjoner basert på forskjellige dimensjoner. Greenberg et al. omtaler fem målbare dimensjoner av proxemics: *avstanden* mellom enheter og brukere, *retning* gir et mål på vinkelen mellom enheter og brukere[3], *bevegelse* omfatter forandring i distanse over tid, *identitet* beskriver en enhet i et gitt detaljnivå og *plassering* beskriver posisjonen til enheten. Proxemics kan i sin helhet være med på å tilby en mer naturlig interaksjon med enhetene våre og kan kanskje være det neste steget for utviklingen av interaksjonsapplikasjoner.

Gester

Interaksjon gjennom gester er et forsøk på å tilby en mer naturlig interaksjon med datamaskinen. Kroppsspråket er tross alt en stor del av mellommenneskelig kommunikasjon, så hvorfor ikke forsøke å utvide det til kommunikasjon mellom menneske og maskin. De fleste prosjektene som utforsker bruker av gester benytter kameraer som måler farger og dybde, samt avanserte datasynsalgoritmer for å gjøre mening av dataene. Sammen kan disse prosessere informasjonen og danne grunnlaget for et system som kan gjenkjenne kompliserte gester. Et eksempel omtales av Dixon et al., der *Kinect*-kameraet[4] ble brukt til å forstå gester.

Taylor et al. (2014) hos Microsoft Research har utviklet en keyboard-prototype som forstår enkle gester. Prototypen forsøker å danne en bro mellom touch-grensesnitt og tradisjonelle keyboard. Keyboardet har 64 sensorer parvis plassert mellom tastene, der 32 sender et IR-signal og de andre 32 sanser reflekterte signaler. Av disse dataene dannes det tre bilder: et av rådataene, et som viser historie over nærhet og et med historie over bevegelse. Prosjektet brukte *random forests* for å klassifisere gestene. Bruksområdet for keyboardet er å bevege seg mellom applikasjoner, zoome eller scrolle, uten å måtte ta hendene vekk fra keyboardet.

Multimodalitet

Multimodale brukergrensesnitt analyserer og forstår flere kommunikasjonskanaler samtidig. Dette kan gi systemer som er mer naturlige å interagere med. For eksempel kan mennesker

[3] Det finnes allerede enheter som gjør bruk av retning, for eksempel ved å skru seg av når folk ikke ser på dem.

[4] https://www.microsoft.com/en-us/kinectforwindows/develop/

kommunisere med datamaskinen gjennom tale og kroppsspråk. Å tilby flere kommunikasjonskanaler til brukeren kan gjøre systemet mer robust, ved å kombinere delvis informasjon fra flere kilder. Multimodalitet kan også gi brukeren muligheten til å selv velge den foretrukne modaliteten. Etter at inputsignalene har nådd systemet må disse analyseres og forstås. Dette steget kalles *fusion*, og omhandler integrasjonen av modalitetene. Det finnes ulike detaljnivåer å integrere signalene på. Et høyere semantisk nivå av fusion er når signalene først forstås hver for seg, og deretter integreres. Alternativt kan datastrømmene integreres tidlig, før de forstås. Dumas et al. (2009) omtaler de ulike formene for fusion som *data-level*, *feature-level* og *decision-level* Jaimes and Sebe (2007) kaller de samme kategoriene *early*, *intermediate* og *late fusion techniques*. Late/decision-level fusion er den mest brukte teknikken. Den involverer ofte uavhengige klassifikatorer for hver inputstrøm. Deretter kombineres disse videre til en enkelt, felles klassifikator. Ettersom kildene datastrømmene håndteres uavhengig trenger de ikke å utføres samtidig. Et multimodalt system kan også ha flere modaliteter for å gi feedback. *Fission* er når systemet avgjør hvilke modaliteter som skal benyttes for å kommunisere tilbake til brukeren.

Hjerne-maskin-grensesnitt

Hjerne-maskin-grensesnitt (BCI) er teknologi for å la hjernen kommunisere direkte med maskinen. Forskningen rundt denne teknologien dreier seg hovedsaklig om å hjelpe eller forbedre funksjonshemmede mennesker. Mennesker med motorkontrollen i orden bruker kroppen sin som et grensesnitt til verden. Dette systemet har utviklet seg over millioner av år og fungerer utmerket. Brukergrensesnitt til datamaskiner er et helt nytt konsept. Dersom det er problemer mellom kroppene våre og datamaskiner, ville det ikke vært mest naturlig å sette spørsmål ved datamaskinen, ikke kroppen? Datamaskinene bør vel tilpasses til å best mulig bruke våre kropper, i stedet for å forbigå kroppen totalt. Vi er allerede i ferd med å miste kroppene våre. Vi sitter stille mesteparten av dagen, både gjennom arbeid, reise og fritid. Dette er teknologi som er verdt å utforske for å hjelpe funksjonshemmede. Det er også interessant å se hva datamaskinene kan lære om følelsene våre ved å lytte på hjerneaktiviteten. Men jeg er usikker på om BCI er verdifullt å utforske og om det er det vi ønsker for framtidens

friske mennesker.

Dynamiske brukergrensesnitt

I kapittel 2.1 så vi på hva brukerene ønsker seg. Noen av disse ønskene kan være vanskelig å implementere uten å bryte noen av de andre ønskene. For eksempel som at systemet skal forstå konteksten i omgivelsene, uten å være påtrengende og oppleves som overvåkende. Det viktigste for de spurte gruppene var å selv være i kontroll av et system som var enkelt å bruke og som ga muligheter til å styre huset, inkludert elektriske apparater, hvitevarer, hjemmeunderholdning, lys og varme. Det er kanskje viktigere for vanlige brukere at det tilbys gode styringsmuligheter, enn at systemet er proaktivt og forsøker å forstå hva brukeren ønsker?

Schneiderman and Maes (1997) diskuterer om direkte manipulasjon er det beste, eller om brukergrensesnitt skal tilpasse seg dynamisk og forsøke å hjelpe brukeren. Schneiderman argumenterer for at vi bør utnytte øyets formidable kapasitet og anvende langt *flere* ikoner, knapper og vinduer enn det som er vanlig i dag. 4000 eller flere elementer på skjermen gir brukeren en full oversikt over mulige valg. Oversikt er det viktigste. Brukeren kan deretter zoome inn på det de er ute etter og filtrere ut det de ikke er interessert i. Dette gir brukeren en følelse av kontroll og dermed en ansvarsfølelse til avgjørelsene de tar. Maes argumenterer for verdien av programvareagenter; programvare som kan ta egne avgjørelser. Grunnlaget for å ønske dette er datamaskinmiljøer som øker i kompleksitet og en brukermasse som øker i naivitet. Når det blir for mye å holde styr på må oppgaver delegeres og programvareagenter kan da være til hjelp. Vi trenger "ekstra øyne" til å finne informasjon vi kan være interessert i. Et eksempel på en slik agent er anbefalingssystemer, som når *Netflix* anbefaler filmer basert på hva brukeren og andre liknende brukere har sett på[5]. Eller *Amazon*, som foreslår varer basert på hva andre brukere som kjøpte den samme varen også viste interesse i. Det kan være stor verdi i at systemet foreslår ting brukeren aldri ville tenkt på eller kommet borti[6]. Maes er enig i at det trengs godt designede grensesnitt for manipulasjon, men at det er en rekke oppgaver brukeren ikke har lyst til å utføre og at programvareagenter kan ta seg

[5] https://www.netflix.com/no/
[6] http://www.amazon.com/

av disse oppgavene.

Rogers (2006) påpeker at forskningsmiljøet har møtt problemer med å forstå den store variasjonen i hva folk gjør, hvilke motiver de har, når de gjør det og hvordan de gjør det. Konteksten rundt folks dagligdagse liv er svært subtil. Dette gjør det vanskelig, om ikke umulig, å implementere kontekst slik at det kan gjøres nyttige spådommer om hva mennesker føler, hva de ønsker, eller hva de trenger, i et gitt øyeblikk. Hun argumenterer for et skifte fra proaktiv databehandling til proaktiv mennesker. Allestedsnærværende teknologier bør designes slik at brukere kan interagere med dem mer aktivt. Mennesker, i stedet for maskiner, bør ta initiativet til å være konstruktive, kreative og engasjere i interaksjoner. I stedet for å pakke omgivelsene med all slags forsvinnende teknologi, bør vi tenke på teknologien som et sett med verktøy og overflater, som er mobile og tilrettelegger for samarbeid. Informasjonskilder, andre mennesker og omgivelsene kan sees og interageres med når det er behov for det. Vi må få tilbake gleden med interaksjon på innovative måter.

Interaktiv media gir brukere personlig tilgang til informasjon. Ishizaki (1996) presenterer *maDes*, en arkitektur som oppfordrer designere til å anse et *designproblem* som en kontinuerlig strøm, framfor en samling diskret problemer, og en *designløsning* som en entitet generert av den dynamiske aktiviteten til *designelementer*, framfor et sett med designelementer med faste attributter. *maDes* modellerer en designløsning som et system bestående av en samling av mindre designsystemer. Hvert system er en *designagent* og er ansvarlig for å presentere en bestemt type informasjon. En designagent kan tilpasse hvordan informasjon vises ut fra forandring i kontekst og gjennom samarbeidet med andre designagenter. I designet av programvare-basert media opplever designeren at det er umulig å designe en løsning til et spesifikt problem og at han i stedet må representere en måte å designe programvare på som kan generere løsninger mens programmet kjører.

Tale

Å gi datamaskinen evnen til å forstå og utøve tale er en av teknikkene fra AI som har hatt kommersiell suksess. Talegjenkjenning benyttes daglig av brukere verden over for å gi navigasjonsinstrukser til bilene sine, gjøre søk på nettet eller å skrive gjennom diksjon. Tale

er en attraktivt interaksjonsform i alle tilfeller der brukeren kan ha bruk for å gjøre andre ting med hendene, eller der han ikke har muligheten til å bruke dem. Det er ingen enkel oppgave å gjenkjenne tale. Lydene brukeren lager inneholder ofte støy og det finnes en rekke setninger som høres like ut når de sies fort. Når vi skriver setninger er det mellomrom mellom ordene, men i tale er det setninger som uttales uten pause mellom ordene. Det er også mange ord som uttales likt, men skrives forskjellig og har forskjellig betydning avhengig av kontekst. Carnegie Mellon University påpeker at virkeligheten med tale ikke er så enkelt som man skulle tro. Intuisjonen er at tale er bygget opp av ord, og at hvert ord består av fonemer. I virkeligheten er det desverre svært annerledes. Tale er en dynamisk prosess uten klare skiller mellom ulike deler. De moderne måtene å beskrive tale på er i større eller mindre grad basert på sannsynligheter. Denne måten å se situasjonen på lar oss forstå at talegjenkjenning aldri vil være 100% korrekt[7]. Den vanlige måten å gjenkjenne tale på er følgende: man mottar en bølgeform, splitter den inn i ytringer delt av stillhet og forsøker så å gjenkjenne hva som blir sagt i hver ytring. For å gjøre dette tar vi alle mulige kombinasjoner av ord og forsøker å matche dem med lyden. Vi velger så den kombinasjonen som passer best.

Tre modeller benyttes i talegjenkjenning for å matche ord med lyd. En akustisk modell, en ordbok og en språkmodell. Den akustiske modellen holder akustiske egenskaper for kombinasjoner av fonemer. En enkel fonetisk ordbok knytter ord med fonemer. Denne naive varianten er ikke veldig effektiv ettersom den ikke tar større hensyn til forskjellige uttaler, men den fungerer som regel i praksis. Maskinlæring kan benyttes her for å lære langt mer nyanserte sammenhenger. En språkmodell brukes for å innsnevre søket etter det passende ordet. Den definerer hvilke ord som kan følge etter det foregående gjenkjente ordet og hjelper dermed med å fjerne ord som ikke er sannsynlige. For å oppnå en god presisjon må språkmodellen utføre en god jobb på å begrense søkeområdet for mulige ord. Den må altså være svært god på å gjette det neste ordet. En språkmodell kan for eksempel si at setningen "Skru av lyset" er 1000 ganger så sannsynlig som "Skru av fryse". Russell and Norvig (2010) omtaler talegjenkjenning som å være identifikasjonen av en sekvens av ord ut i fra et akustisk signal. Talegjenkjenning handler om å finne den mest sannsynlige sekvensen av ord. De fleste talegjenkjenningssystemene i dag benytter en språkmodell etter *Markov-antakelsen*. Dette

[7] http://cmusphinx.sourceforge.net/wiki/tutorialconcepts

betyr at det nåværende ordet kun avhenger av et visst antall foregående tilstander. Det nåværende ordet representeres gjerne som en enkelt, tilfeldig variabel. Dette gjør modellen i sin helhet til en *Hidden Markov Model (HMM)*. Modellen beskriver en sekvens av tilstander der overgangen er gitt av en sannsynlighet. Det er en generell modell som kan beskrive alle sekvensielle prosesser og den har vist seg å være spesielt praktisk for å dekode tale.

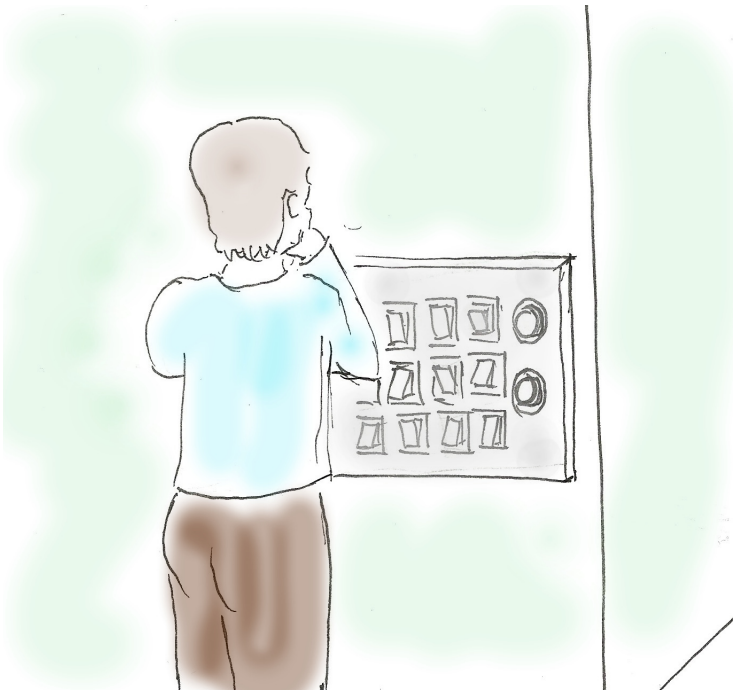
3 Metoder & Implementasjon

3.1 Min subjektivitet

Gjennom introduksjonen og bakgrunnsteorien har jeg allerede omtalt noen av mine egne tanker rundt hva slags programvare som er aktuell å utforske i smarte hjem. Noen av disse idéene hadde jeg allerede når jeg begynte på denne oppgaven. Jeg visste at jeg ville utforske lineære klassifiseringsalgoritmer og kontekstdrevne brukergrensesnitt. Dette utgangspunktet har gjort at jeg kan ha unngått å se mulige innfallsvinkler til oppgaven. At jeg har hatt flere idéer jeg ville undersøke har også gjort at jeg har hatt for liten tid til å gå så dypt i hver idé som jeg kunne ønsket. Forslag til videre arbeid presenteres i kapittel 5.2. Denne subjektiviteten har sannsynligvis hatt en påvirkning på de følgende hypotesene.

3.2 Hypoteser

Når hjemmet ditt om noen år tilbyr kontroll over ikke bare lys og temperatur, men garasjeporter, persienner, tv-er, radioer, låsene på døra og statusen til kjøkkenapparater, kan det være en utfordring å tilby gode interaksjonsmetoder. Løsningen på dette har hittil enten vært å la kontrollknappene være en del av apparatet, eller å samle de i et panel på veggen, i en fjernkontroll eller i en app. Med *Tingenes Internett* (IoT) og et økende antall styrbare enheter blir det raskt upraktisk å kun ha kontroll dersom man fysisk befinner seg ved apparatet. Det virke fornuftig å dermed tilby kontroll gjennom en fjernkontroll eller en app. Men vil man alltid ha kontroll på hvor denne mobile enheten befinner seg? En fjernkontroll eller app må også designes svært godt for å være oversiktlig. Vi har alle vært uerfarne brukere av en ny fjernkontroll og opplevd større eller mindre problemer med å utøve kontroll over det aktuelle apparatet. Vi bør ikke slå oss til ro med å gjenskape et stort knappepanel i en app og kalle det et optimalt design. Så kanskje det ikke er en dum idé å tilby et fast sted i rommet der kontrollen over aktuelle enheter er samlet? Det tradisjonelle panelet med knapper og dimmere tar ikke bare stor plass, men det er i tillegg vanskelig å vite hvilke knapper som hører til hvilken funksjonalitet (se figur 5). Det ideelle hadde kanskje vært å



Figur 5: Stort panel med knapper skaper forvirring.

tilby et fast sted i rommet der kontroll kan utføres, men som er minimalistisk og kan styre et stort antall enheter. Hva hvis brukere kunne utføre enkle gester i luften foran en svært liten sensor, strategisk plassert i rommet?

Hypotese 1 - Gestegjenkjennelse gjennom fotodioder

En enkel sensor med fotodioder kan benyttes som en multifunksjonell, mekanisk bryter, og bruk av maskinlæring kan gi sensoren evnen til å skille mellom flere kommandoer enn eksplisitt programmering kan.

Programvare for å forstå tale blir stadig bedre. Desverre er det et stort problem å lage et brukbart system i scenarier der programvaren skal lytte etter kontinuerlig tale og må kunne skille mellom hva som er en kommando, og hva som kun er ordinær tale mellom mennesker. Det ville for eksempel vært et problem dersom alle som kom hjem på besøk måtte få en innføring i alle ord som ville utløse kommandoer og som de ikke kunne bruke mens de var i huset. Kommersielle systemer for talegjenkjenning løser gjerne dette ved å lytte etter et kodeord, men med denne tilnærmingen forsvinner drømmen om å ha en kontinuerlig dialog

med datamaskinen. Forståelse av naturlig språk har et annet stort problem; prosesseringen og forståelsen av taledata må foregå i kraftige serverparker langt unna brukeren. Dette er et problem med tanke på personvern og det er et praktisk problem med en ikke ubetydelig ventepause før resultatet returneres til brukeren. Kanskje det ikke er nødvendig å forstå naturlig, kontinuerlig tale for å tilby kontroll over hjemmet?

Hypotese 2 - Multimodal interaksjon gjennom tale og gester

Kombinasjonen av enkle gester og begrenset tale er en tilstrekkelig, naturlig og effektiv måte å kontrollere hjemmet på.

En gest utført foran flere sensorer i en viss konfigurasjon vil skape mer data. Det virker derfor rimelig å anta at mer data kan lede til en enda mer nøyaktig forståelse av ulike kommandoer. Sensorene har også muligheten til å måle andre verdier, som lys-nivåer, nærhet og tilstedeværelse. Kan disse egenskapene utnyttes i et smart hjem?

Hypotese 3 - Kombinasjoner

Ved å benytte flere sensorer i kombinasjon kan man oppnå en høyere presisjon enn ved bruk av en enkelt sensor, flere kommandoer kan utføres naturlig og sensorenes evne til å måle lys, nærhet og tilstedeværelse kan også utnyttes verdifullt i et smart hjem.

Standard programvare for styring av smarte hjem er i dag grafiske brukergrensesnitt der metaforiske objekter, som knapper og brytere skal manipuleres. Er dette egentlig nødvendig? Brukerne bryr seg ikke om disse kunstige objektene. De bryr seg om informasjonen om hjemmet og om å forstå valgene de kan ta. Den eneste modellen som skal manipuleres befinner seg i hodene deres. Et grafisk brukergrensesnitt for smarte hjem er informasjonsprogramvare og bør designes som en informasjonsgrafikk og ikke som manipulasjonsprogramvare.

Hypotese 4 - Kontekstdrevet brukergrensesnitt

Et grafisk brukergrensesnitt for det smarte hjemmet bør være kontekstsensitivt og være drevet av data, ikke interaksjon med brukeren.

3.3 Design av eksperimenter

Gestegjenkjennelse gjennom fotodioder

For å teste denne hypotesen må det først finnes fram en aktuell sensor og deretter etablere at den kan benyttes som en multifunksjonell mekanisk bryter. Med dette grunnlaget kan man utforske dataene sensoren produserer. Deretter kan dataene prosesseres og programvare kan utvikles for å lære forskjellene i dataene ulike gester produserer. Resultatene fra et slikt system må være tilsvarende eller bedre enn resultatene fra et eksplisitt programmert system. Det er to dimensjoner til dette resultatet: antallet ulike gester systemet kan skille mellom og hvor ofte gestene forstås korrekt.

For å gjennomføre eksperimentet trengs det tilgang til en sensor som produserer tilstrekkelig detaljert data når et objekt føres i nærheten. Dataene må overføres fra sensoren til en kraftigere datamaskin. Her må dataene behandles og forberedes til maskinlæring. Til slutt trengs det gode biblioteker for læringen, gjerne med en rekke tilgjengelige algoritmer så flere tilnærminger kan forsøkes. Maskinlæringsalgoritmer er sjeldent svært avanserte og vanskelige å implementere, men for å oppnå gode resultater hurtig trengs optimaliserte implementasjoner. Det virker derfor rimelig å lete etter passende biblioteker, framfor å implementere egne algoritmer.

Multimodal interaksjon gjennom tale og gester

I mangelen på tilgang til empirisk brukertesting kan denne hypotesen utfordres med argumentasjon. Først må det etableres at dagens løsninger for kontinuerlige tale har problemer med personvern og responstid. Videre må det argumenteres for at enkle gester og begrenset tale fungerer godt til å styre hjemmet. Til slutt må det vises hvordan eventuell multimodal input kan håndteres og gi tilfredsstillende resultater.

Dersom eksperiment 1 er en suksess er gestegjenkjennelse tilgjengelig. Videre må det velges eller implementeres et system for talegjenkjenning. Det trengs i tillegg til utstyret fra eksperiment 1 en mikrofon for å lytte etter tale. Det trengs også teknikker for å håndtere data fra gester og tale samtidig. Å håndtere simultan data fra både gester og tale er et problem. Hvordan sikrer man at data ikke går tapt dersom input kommer samtidig fra begge

kilder? Hvilken data skal ha prioritet dersom de kommer samtidig? Til sist bør eksperimentet visualiseres på et vis for å vise at systemet fungerer slik som ønsket.

Kombinasjoner

Maskinlæring må benyttes igjen i dette eksperimentet og bruken av flere sensorer må gi et betydelig bedre resultat i en eller begge av dimensjonene jeg har nevnt. Deretter må det kunne vises at de andre egenskapene ulike enkle sensorer har kan benyttes produktivt i forbindelse med et smart hjem. Dette kan kun argumenteres.

Jeg tar utgangspunkt i at eksperiment 1 og 2 har vært vellykkede. Det trengs flere sensorer, og eventuelt annen hardware for å håndtere disse. Posisjonen til sensorene er interessant. Skal de plasseres så nære hverandre som mulig, for å fungere som en enkelt bryter med større presisjon? Eller skal de posisjoneres et godt stykke fra hverandre, så de kan benyttes med begge hender og som en enkelt eller delvis flere brytere?

Kontekstdrevet brukergrensesnitt

Denne siste hypotesen testes ved å implementere et grafisk brukergrensesnitt og vise hvordan det i dette scenariet er overlegent tradisjonelle, interaksjonsdrevne systemer. Hypotesen omhandler informasjonsprogramvare; hvordan relevant informasjon om hjemmet kan presenteres for brukeren. For å implementere eksperimentet må det velges teknologi for å lage et brukergrensesnitt. Aktuelle muligheter er webteknologi, animasjon av et slag, eller programvare til desktop eller app-er. Det trengs teknologi med evnen til å forandre seg basert på data og ikke nødvendigvis interaksjon med brukeren.

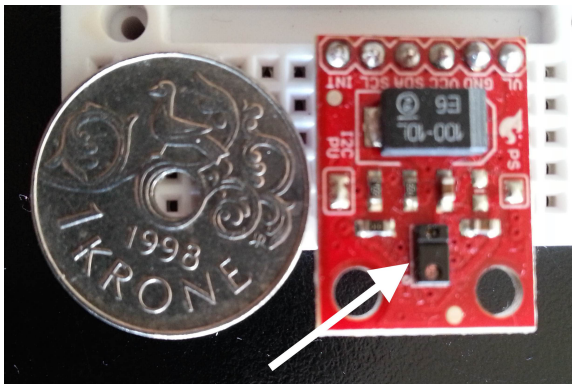
3.4 Implementasjon

3.4.1 Gestegjenkjennelse gjennom fotodioder

For å implementere eksperimentet har jeg brukt sensoren *APDS-9960* fra Avago Technologies[8]. Denne sensoren har flere funksjoner og tilbyr måling av lys og farge, oppdagelse av

[8] http://www.farnell.com/datasheets/1801560.pdf

nærhet og gestegjenkjennelse. Innpakningen er svært liten på kun 3.94 * 2.36 * 1.35 mm, og kan ses i figur 6. Gestesensoren selv består av fire fotodioder, som kan oppfatte et infrarødt signal. En LED sender ut det infrarøde signalet og dersom et objekt befinner seg innenfor omtrent 20 cm vil signalet reflekteres tilbake med nok styrke til å bli oppfattet av fotodiodene. Fotodiodene er vinklet litt forskjellig slik at de plukker opp refleksjoner fra ulike retninger. Gestesensoren benytter resultater fra en nærhetsdetektor for å automatisk aktiveres når et objekt er innen rekkevidde. I tilegg brukes måling av lys for å tilpasse de infrarøde målingene til lysnivået i omgivelsene. Disse egenskapene gjør denne sensoren mer nøyaktig enn enklere varianter som benytter én LED og én fotodiode (som keyboardet beskrevet av Taylor et al. (2014)). Dataene dannes som 32-bit datasett og kommuniseres over I2C-protokollen til en mikrokontroller. For å utvikle systemet er *Sparkfun*'s innpakning av APDS-9960-sensoren benyttet[9]. Figur 6 viser APDS-9960-sensoren på Sparkfun-brettet. Dette brettet gjør sensoren tilgjengelig for enklere prototyping ved å bryte ut ulike pinner. I tillegg har Sparkfun skrevet programvare til *Arduino*-plattformen[10] så utviklere kommer raskt i gang og kan gjøre bruk av de forskjellige funksjonalitetene hos APDS-9960-brikken.



Figur 6: Sensorstørrelse: APDS-9960, montert på Sparkfun-brettet.

For å trene systemet kreves data og for å få dataene inn til datamaskin kreves en mikrokontroller. Med Sparkfuns programvare for Arduino var det naturlig å velge en Arduino som mikrokontroller. Arduinoen ble koblet til sensoren ved å følge oppkoblingsguiden på hjemmesidene til Sparkfun[11]. Gestesensoren drives på 3.3V, mens en normal Arduino Uno drives på

[9] https://www.sparkfun.com/products/12787
[10] http://www.arduino.cc/
[11] https://learn.sparkfun.com/tutorials/

Tabell 1: Utstyrsliste eksperiment 1

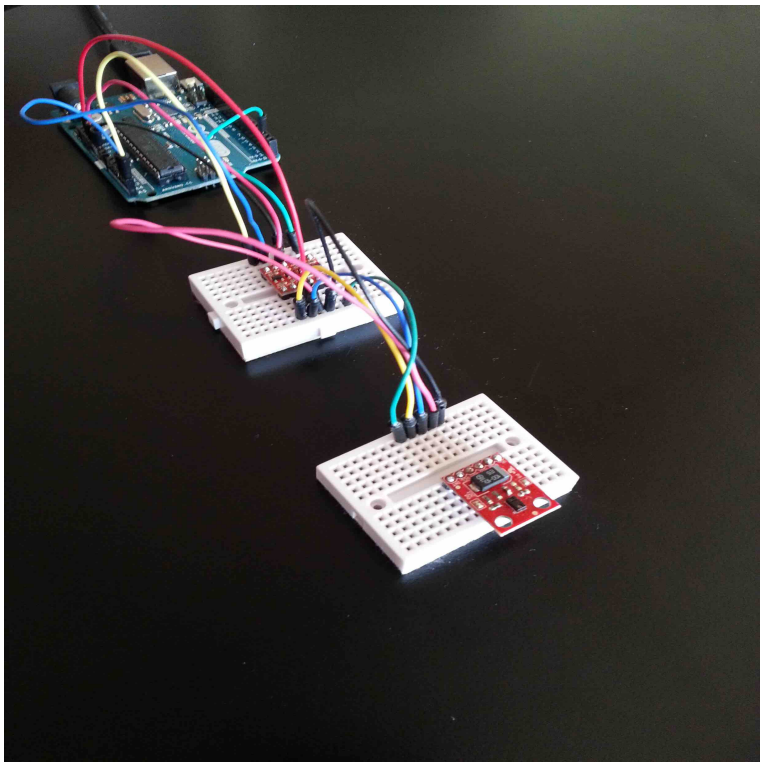
Utstyr	Bruksområde
Sparkfun APDS-9960	Gestesensor
Level-shifter	Konverterer mellom 5V og 3.3V
Breadboard	Montere sensor og level-shifter for enklere prototyping
Break-away headers	Montere sensor og level-shifter til breadboard
Loddebolt og tinn	Lodde headers til sensor og level-shifter
Ledninger	Overføre signal mellom sensor, level-shifter og Arduino
Arduino Uno	Drive sensoren og sende data til datamaskinen
USB-kabel	Overføre data til datamaskinen
Macbook Pro	Håndtere data og utføre maskinlæring

5V. Dermed ble det nødvendig å introdusere en level-shifter, for å konvertere fra 5V til 3.3V. Tabell 1 viser eksperimentets utstyrsliste. Figur 7 viser hvordan komponentene ser ut etter å ha blitt koblet sammen og er klargjort for å utføre eksperimentet.

Etter at Arduinoen, sensoren og datamaskinen har fått kontakt kan jeg benytte programvaren til Arduino-plattformen til å laste opp koden som trengs for å drive sensoren. Jeg tilpasset biblioteket fra SparkFun minimalt, slik at Arduinoen ikke selv forsøker å forstå sensordataene, men i stedet sender dataene videre til datamaskinen. For å håndtere dataprosesseringen og utføre maskinlæring valgte jeg programmeringsspråket *Python*[12]. Dette valget ble tatt med utgangspunkt i at Python har et bredt utvalg biblioteker innen både seriell kommunikasjon og maskinlæring. Jeg vurderte noen alternativer og gjorde noen utforskende forsøk med å benytte Java-plattformen og et av språkene som kjører på JVM. Java har mange gode biblioteker, spesielt innen maskinlæring. Desverre støtte jeg på problemer med å håndtere den serielle kommunikasjonen. Jeg antar at bruken av en virtuell maskin skaper noe mer kompleksitet rundt å kommunisere via de noe utdaterte serielle portene. Et siste alternativ jeg vurderte var å benytte programmeringspråkene Matlab eller R, som begge er populære verktøy innen AI. Igjen viste det seg at Python hadde et fortrinn når det gjaldt den serielle kommunikasjonen.

Arduino-programvaren henter data via I2C-protokollen fra sensoren. Den sender så dataene videre serielt til datamaskinen som lytter på den aktuelle porten. Sensoren sender data så lenge det infrarøde signalet reflekteres og oppfattes av fotodiodene. Dette betyr at

[12] http://www.python.org



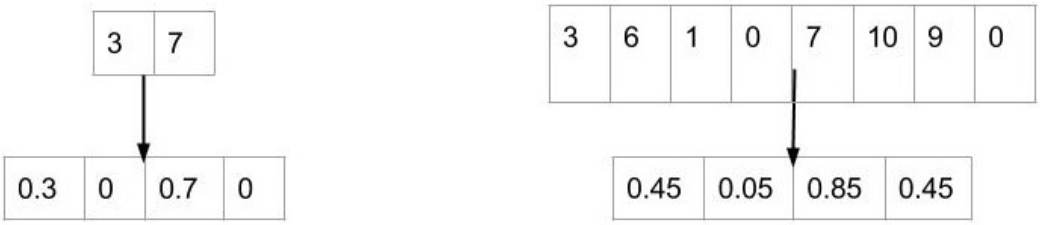
Figur 7: Sensor, level shifter og Arduino.

en gest som tar lengre tid skaper mer data. Et raskt flikk med to fingre skaper 16-32 datapunkter. Et rolig sveip over sensoren med hele hånda kan skape 100-200 datapunkter. En gest som involverer å holde hånda foran sensoren i flere sekunder skaper hundrevis av datapunkter. Den variable mengden datapunkter skaper et problem; for å benytte de planlagte klassifiseringsteknikkene må hvert av treningseksemplene må ha like mange datapunkter.

Det finnes ulike metoder for å løse dette problemet. En løsning er å bestemme det maksimale antallet datapunkter som skal tas med. Inputeksempler som ikke har tilstrekkelig med datapunkter får lagt til 0-verdier for å oppnå den ønskede størrelsen. Å sette en slik maksgrense på datapunkter kan desverre føre til at man mister viktig data fra gester som tar lengre tid å utføre. Og for et inputeksempel fra en rask gest vil mange datapunkter være 0. Disse problemene kan påvirke effektiviteten til læringsalgoritmen. Et annet alternativ er å velge et fast antall datapunkter vært inputeksempel skal ha. Deretter knytter man inputeksempelet til denne vektoren av fast størrelse. Dersom inputdataene har få datapunkter blir den resulterende vektoren sparsom, med dataverdiene spredt jevnt utover, og med 0-verdier i mellom. Dersom inputdataene består av mange datapunkter vil hvert datapunkt i

vektoren være et gjennomsnitt av en valgt mengde datapunkter.

Jeg valgte å benytte denne sistnevnte teknikken og lagde vektorer med 128 datapunkter. Dette tallet ble valgt basert på tester av ulike aktuelle gester og antallet datapunkter disse genererte. 128 datapunkter er nok til å gi tilstrekkelig detaljer selv ved gester som tar noe lengre tid og samtidig ikke så mange at raske gester skaper i overkant sparsomme vektorer. Vektoren normaliseres ved å knytte de mulige sensorverdiene [0,255] til [0.0,1.0]. Å illustrere dannelsen av vektorer med 128 datapunkter blir tungvint så i figur 8 har jeg illustrert prosessen med langt mindre data. I 8a består inputvektoren av to datapunkter. La oss si vi ønsker en vektor med størrelse fire og som er normalisert til verdiområdet [0.0,10.0]. Dette oppnås ved å spre inputdataene jevnt i vektoren og normalisere verdiene. 8b viser det motsatte tilfellet, der inputvektoren er for stor. For å representere dataene i en mindre vektor blir det tatt gjennomsnittsverdier som deretter normaliseres. Listing 1 viser koden som



(a) (b)

Figur 8: Dannelsen av en datavektorer.

prosesserer dataene og forbereder dem på maskinlæringen. På linje 3 dannes det en vektor med 128 0-verdier. Linje 4 avgjør om inputdataene skal spres jevnt i resultatsvektoren eller om det skal dannes gjennomsnittsverdier.

Eksperimentet vil ha to deler. Den første delen blir å utføre en rekke eksempler på gester og lagre de prosesserte dataene. Del to er å laste inn dataene og danne en modell som kan trenes, testes og evalueres. For å utføre maskinlæringen benyttet jeg to Python-biblioteker:

```
def preprocess_data(d, n=128, m=255):
    l = len(d)
    result = [0] * n
    s = int(l / n)
    if s == 0:
        s = int(1. / (float(l) / float(n)))
        for i, x in enumerate(d):
            result[i*s] = x/m
    else:
        for i in range(0, n):
            result[i] = sum(d[i*s:(i+1)*s])/(m*s)
    return result
```

Listing 1: Dataprosessering

NumPy[13] og *scikit-learn*[14]. Numpy er det mest brukte biblioteket for numeriske kalkulasjoner i Python. For maskinlæringen finnes det mange alternativer. Scikit-learn ble valgt fordi det tilbyr et stort utvalg algoritmer og er spesielt gode på de lineære klassifiseringsalgoritmene. For å benytte klassifiseringsalgoritmene scikit-learn har tilgjengelig trengs det to NumPy-datasett: et datasett med dataene fra gestene, og et tilsvarende med tilhørende klasser. Listing 2 benytter *kryssvalidering* til å splitte dataene (X) og klassene (y) inn i trenings- og

```
X_train, X_test, y_train, y_test
    = cross_validation
        .train_test_split(X, y, test_size=0.25, random_state=r.randint(0, len(X)))
```

Listing 2: Splitte datasettene

testsett. Jeg valgte å dele datasettene i 75% til trening og 25% til testing.

3.4.2 Multimodal interaksjon gjennom tale og gester

Det finnes alternativer dersom man ønsker å utforske programvare for talegjenkjenning. Blant andre tilbyr *Google* sitt tale-API til interessenter. Min idé er at disse tjenestene muligens ikke er det vi ønsker i hjemmet, så vi må ty til andre midler. Dette eksperimentet handler delvis om å argumentere hvorfor dette er tilfellet og blir greid ut om i kapittel 4.

[13] http://www.numpy.org/
[14] http://scikit-learn.org/stable/

Hva er statusen på talegjenkjenningsteknologi som ikke benytter internettkommunikasjon med kraftige servere? Etter en utforskning av alternativene valgte jeg å benytte Pocketsphinx, en av talegjenkjenningsmotorene fra Carnegie Mellon University's Sphinx-prosjekt[15]. *CMU Sphinx* er et pågående, åpent prosjekt og de har to aktuelle prosjekter for talegjenkjenning: *Sphinx4* og *Pocketsphinx*. Pocketsphinx er designet for mobile enheter og andre alternative, ressursbegrensede plattformer. Den har et konfigurerbart vokabular så man kan skape modellene man trenger. I følge en nylig skrevet artikkel kan Pocketsphinx med god konfigurering gjenkjenne 10000 ord i reelltid med en feilrate på omtrent 20%[16]. Dette er teknisk imponerende for denne ressurseffektive programvaren, men det er en for stor feilrate til å være praktisk i kommersiell bruk. Heldigvis øker presisjonen betydelig med et mindre vokabular og med under 100 ord er feilraten 3%. Pocketsphinx tilbyr også å benytte et aktiveringsord, slik at enheten kan lytte kontinuerlig gjennom dagen.

Sphinx4 er skrevet i Java og ble utforsket i forbindelse med å utføre både gesteforståelse og talegjenkjenning på Java-plattformen. Pocketsphinx er skrevet i *C* og er dermed lettere tilgjengelig for bruk gjennom Python. Etter at jeg hadde avgjort å jobbe med Python ble Pocketsphinx det naturlige valget. For å benytte Pocketsphinx trengs det tre datakilder: en HMM, en språkmodell og en ordbok. CMU Sphinx-prosjektet utvikler stadig nye HMM-modeller og den engelske er godt moden. Desverre har de ikke jobbet med en norsk modell. Det finnes modeller for norsk der ute, men da jeg ikke fant noen åpne og fritt tilgjengelige ble jeg tvunget til å utføre eksperimentet på engelsk. Dette var ikke et problem, da eksperimentet handler om argumentasjon for lokal og begrenset talegjenkjenning og den praktiske utførelsen av multimodalitet, ikke språket det tales i. Språkmodellen og ordboken er basert på en liste over ord eller setninger som talegjenkjenningen skal forsøke å forstå lyder som. CMU Sphinx tilbyr et verktøy[17] for å lage en språkmodell og ordbok, basert på en liste over aktuelle setninger. Modellen som dannes kan forstå kombinasjoner av ordene og setningene, men kombinasjonen man faktisk skriver inn i lista blir vektet tyngre.

For å håndtere simultan input fra begge datakilder valgte jeg å benytte tråder og en felles FIFO-kø. Hovedtråden i programmet starter to nye tråder som hver har ansvar for å

[15] http://cmusphinx.sourceforge.net/
[16] http://cmusphinx.sourceforge.net/2015/02/
[17] http://www.speech.cs.cmu.edu/tools/

drive input-funksjonene til gestedata og taledata. Listing 3 definerer en funksjon som tar en

```
from threading import Thread

def start_thread(fn, args):
    worker = Thread(target=fn, args=args)
    worker.setDaemon(True)
    worker.start()
```

Listing 3: Bakgrunnstråder

funksjon og argumenter som input, og starter en bakgrunnstråd som utfører funksjonen med argumentene. Funksjonene i bakgrunnstrådene henter data fra seriellporten for gestedata, og

```
from serial import Serial

def gesture_recognition(q):
    ser = Serial('/dev/cu.usbmodem1431', 9600)
    buf = []
    while True:
        c = ser.read()
        if c == '\n':
            result = ''.join(buf)
            q.put(('gesture', result), block=True, timeout=1)
            buf = []
        elif c == '\r':
            pass
        else:
            buf.append(c)
```

Listing 4: Overføre data fra gester

fra mikrofonen for taledata. Disse dataene puttes på en felles kø, som en *tuple*: et kodeord, sammen med dataene. På denne måten blir dataene merket slik at forbrukeren av køen kan skille mellom kildene dataene kommer fra. Hovedtråden lytter etter ny data på køen. På denne måten er de to inputkildene likestilte. Listing 4 viser funksjonen som håndterer input fra gestesensoren. *While*-løkken bygger opp en databuffer. Når alle dataene som representerer en gest har ankommet fra sensoren puttes de på køen sammen med en etikett som forteller om datakilden.

Hovedtråden driver en animasjon. Animasjonen utføres av *pyprocessing*[18], et bibliotek som følger animasjonsspråket *Processing*'s konvensjoner. Det første elementet i data-tuplen

[18] https://code.google.com/p/pyprocessing/

```
import pyprocessing as p

def simple_multi_modal():
    source, recognized_input = animation_q.get_nowait()
    # non-blocking check for items in animation_q.
    # If queue is empty a Queue.Empty exception is raised.
    # data read from queue is a tuple of strings: (source, recognized_input)
    if source == 'gesture':
        p.fill(0, 102, randint(50,150))
        p.textSize(48)
    elif source == 'speech':
        p.fill(102, randint(50,150), 0)
        p.textSize(32)
    else:
        pass
    p.text(recognized_input, randint(50,550), randint(50,550))
```

Listing 5: Håndtere multimodal inputdata

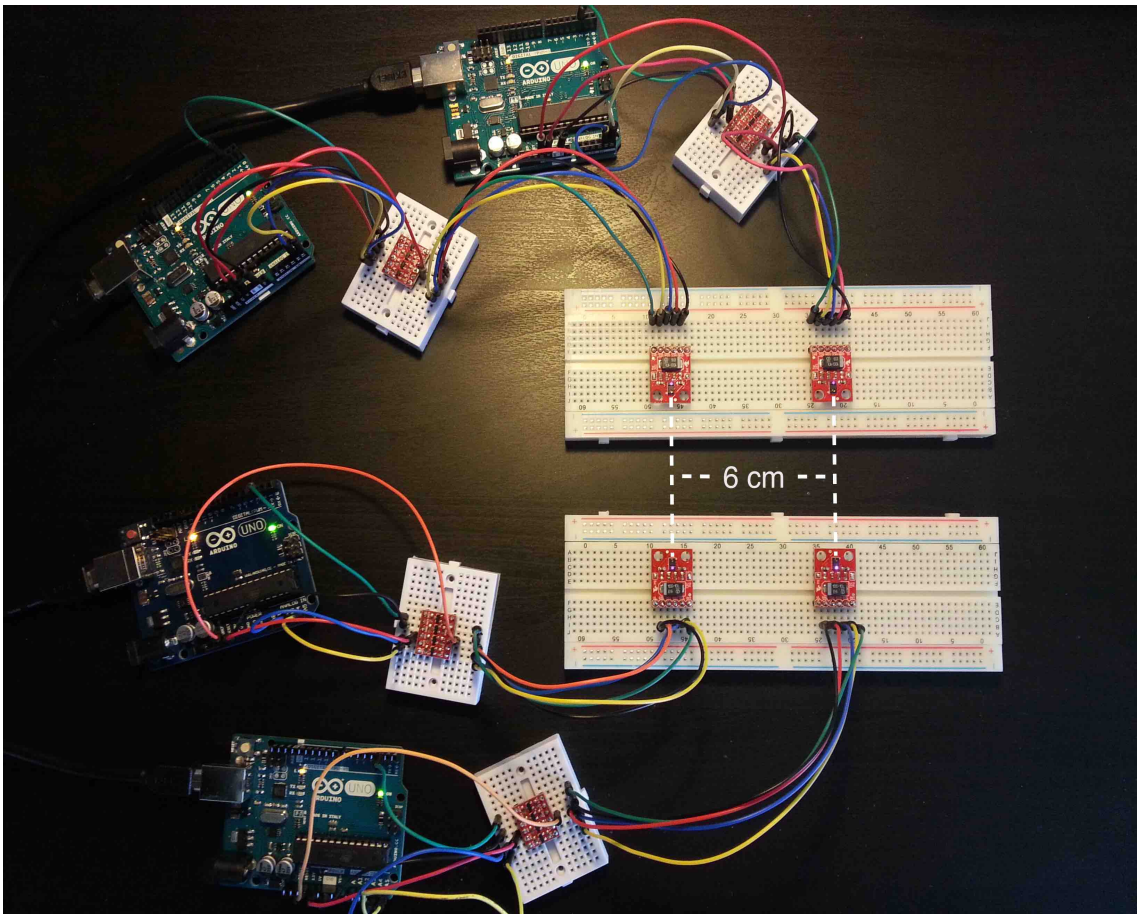
er et kodeord. Dersom kodeordet sier at dataene er gestedata tegnes det på et vis, og dersom dataene er taledata på et annet. Dersom det ikke er ny data i køen tegnes ingenting nytt.

Denne implementasjonen viser hvordan to inputkilder enkelt kan operere asynkront og dytte data inn på en felles kø. Hovedtråden i programmet henter data fra køen og avhengig av kilden tegnes dataene forskjellig på skjermen. Implementasjonen viser et enkelt alternativ til hvordan multimodal input kan håndteres.

3.4.3 Kombinasjoner

Jeg valgte å benytte fire sensorer, der selve sensorene danner hjørner i et kvadrat på 6 * 6 cm, som vist i figur 9. For å håndtere data fra fire sensorer samtidig valgte jeg å benytte samme strategi som i eksperiment 2; fra hovedtråden i programmet starter jeg opp fire tråder som hver lytter på sin port, og dytter data fra gestesensorene inn på en felles kø. Hovedtråden henter data fra køen helt til køen er tom.

De individuelle trådene putter også her dataene på køen sammen med en identifikator, så hovedtråden forstår hvor dataene har sitt opphav. Ettersom dataene fra de ulike sensorene kan komme i en uforutsigbar rekkefølge er jeg nødt til å sortere. Hovedtråden bygger opp lister av data fra hver sensor. Når køen er tom prosesseres rådataene og forberedes til



Figur 9: Fire sensorer.

maskinlæringen. Jeg valgte også her å benytte vektorer med størrelse på 128 datapunkter, for hver sensor. Dette betyr at dersom en gest for eksempel kun blir oppfattet av en av de fire sensorene, vil den resulterende vektoren på 512 datapunkter ha minimum 364 datapunkter som er null. På denne måten vil systemet svært enkelt lære gester som kun aktiverer et subsett av de fire sensorene. Maskinlæringen er tilsvarende i eksperiment 1; scikit-learn håndterer inputdata med vilkårlig store datavektorer.

Listing 6 viser hvordan eksperimentet med nærhetssensoren og dimming kan håndteres. Data fra nærhetssensoren kommer inn til hovedtråden kontinuerlig, men det er først når det riktige ordet ytres og talegjenkjenningen gjenkjenner dette at nærhetsdataene blir brukt. Ved å si "lights" og holde hånda foran sensoren kan brukeren dimme lysnivået.

```
def dimming():
    global lights_activated
    source, light_level = animation_q.get_nowait()
    if source == 'dim' and lights_activated:
        light_level = int(float(light_level))
        if light_level < 200:
            r = light_level
            g = light_level
        else:
            r = 255
            g = 255
        b = light_level - 20
        p.background(200)
        p.fill(r,g,b)
        p.rect(0,0,600,600)
    elif data == 'LIGHTS':
        lights_activated = True
    else:
        pass
```

Listing 6: Dimme lys

3.4.4 Kontekstdrevet brukergrensesnitt

Det var klart at teknologi med av en dynamisk natur burde benyttes i implementasjonen av dette eksperimentet. Det mest dynamiske ville vært animasjonsteknologi, men ettersom brukergrensesnittet bare trenger å oppdatere dersom det er ny data, virket webteknologi som et bedre valg. Jeg valgte å benytte webteknologi *(HTML, CSS, Javascript)*, framfor desktop/app av to grunner: *Javascript*'s dynamiske natur, samt den nokså nye teknologien fra *Facebook* kalt *React*[19]. React tilbyr et virtuelt *DOM* som man programmerer mot. Når data oppdaterer seg beregner React forskjellen mellom dette virtuelle DOM-et og det virkelige DOM-et. Utregningen av denne forskjellen gjør at minimale deler av det virkelige DOM-et må oppdateres, og det hele er svært effektivt. I stedet for å benytte Javascript og React direkte gjorde jeg bruk av *Clojurescript*[20] og *Reagent*[21]. Clojurescript er et programmeringsspråk og en *Lisp*-dialekt, som kompilerer til Javascript. Reagent er et Clojurescript-bibliotek for å bruke React. En fordel ved å benytte Clojurescript er at jeg idiomatisk kan modellere hele hjemmets tilstand i en, enkel datastruktur. Først når hjemmets tilstand forandrer seg

[19] https://facebook.github.io/react/
[20] https://github.com/clojure/clojurescript
[21] https://github.com/reagent-project/reagent

```
(def initial-state
    {:rooms
        {
        ..
        :kitchen
            {:stove {:active? false, :temp nil}
             :oven  {:active? false, :temp nil}
             :dish-washer {:active? false, :time-remaining 45}
             :fridge    ["Eggs" "Bacon" ..]
             :lights-off? false}
        ..
        }
    ..
    :view    :home
    ..
    })
```

Listing 7: Datastruktur

og denne datastrukturen blir oppdatert, vil React automatisk beregne hvilke deler av det grafiske grensesnittet som må oppdateres.

Listing 7 viser deler av datastrukturen. ".." er deler av koden jeg har fjernet for lesbarhet og oversikt. Hjemmets tilstand er representert som et *hash-map*[22]. Nøkkelen *:rooms* har et nytt hash-map som verdi og i dette befinner alle rommene i hjemmet seg. Eksempelrommet *:kitchen* holder data om apparatene og lyset på kjøkkenet. I tillegg til rommene i hjemmet holder datastrukturen annen informasjon om hjemmet, som diagnostikk, vær og brukerpreferanser. Nøkkelen *:view* definerer hvordan det grafiske grensesnittet skal presentere dataene. Datastrukturer i Clojurescript er *immutable*; de kan ikke forandres etter dannelse. Listing

```
(def state (atom initial-state))
```

Listing 8: Atom

8 gjør to ting: den immutable datastrukturen kan nå forandres ved å gå fra en tilstand til en annen, og den er nå koblet til Reacts virtuelle DOM. Listing 9 er hovedfunksjonen for å generere brukergrensesnittet. Funksjonen returnerer et *div*-element, som puttes inn i HTML-dokumentet brukerens nettleser mottar. På linje 5 spørres det om hvilket *view* som brukes. Dette definerer hvilken HTML-genererende funksjon som skal kalles. For eksempel,

[22] En assosiativ datastruktur med konstant tidskompleksitet for lesing og oppdatering

```
1  (defn home-page []
2    (let [view (:view @state)]
3      [:div.container
4       ..
5       (case view
6         :kitchen (kitchen " zoom")
7         :bathroom (bathroom " zoom")
8         :bedroom (bedroom " zoom")
9         :garage (garage " zoom")
10        :hall (hall " zoom")
11        :food (food)
12        :weather (weather)
13        :diagnostics (diagnostics)
14        :home ..
15        )]))
```

Listing 9: Hovedkonteiner HTML

dersom *view* er *:kitchen* ser vi at funksjonen *kitchen* kalles med et strengeparameter. Dette vil generere HTML som definerer kjøkkenet og grensesnittet vil zoome inn så detaljene kan ses bedre. Dersom *view* er *:home* kalles alle funksjonene som genererer rom, men med et tomt strengeparameter.

La oss si at *kitchen* ble kalt. Listing 10 viser denne funksjonen som også returnerer et *div*-element med videre innhold basert på dataene om hjemmets tilstand. For eksempel bestemmer linje 2 hvilke *CSS*-klasser som skal brukes avhengig av om lyset i hjemmet er av eller på. Denne teknikken brukes for å eventuelt skyggelegge *div*-en som representerer kjøkkenet. Linje 5 definerer kjøleskapet som et bilde. Vi ser at dersom dette bildet trykkes på settes *:food* som verdi til nøkkelen *:view*, i hjemmets tilstand. Denne endringen i hjemmets tilstand vil umiddelbart føre til at React beregner hva som må oppdateres. Basert på koden i listing 9 vil ny HTML genereres og brukergrensesnittet vil eventuelt vise dataene annerledes. På linje 7 (listing 10) undersøkes det om oppvaskmaskinen står på eller ikke, og basert på dette kan ulike grafiske representasjoner brukes.

Implementasjonen gjør bruk av et utvalg regler som bestemmer hvordan dataene skal vises. Listing 11 viser en slik regel. Denne regelen sier at når platetoppen er påskrudd og brukeren ikke er hjemme, skal kjøkkenet vises. Denne regelen sikrer at dataene som kan representere en potensielt farlig situasjon blir satt i fokus.

```
(defn kitchen [z]
  [:div {:className (str "room" z (when (:lights-off?
                                         (:livingroom
                                              (:rooms @state)))
                                                  " dark"))}
   [:img.room-image {:src       (:kitchen imgs)
                     :on-click #(swap! state assoc
                                       :view :kitchen)}]
   [:img.fridge {:src       (:fridge imgs)
                 :on-click  #(swap! state assoc
                                    :view :food)}]
   (if (:active? (:dish-washer
                      (:kitchen (:rooms @state))))
     [:div
      [:img.dish {:src      (:dish-on imgs)
                  :on-click #(swap! state assoc-in
            [:rooms :kitchen :dish-washer :active?] false)}]]
      [:img.dish {:src      (:dish-off imgs)
                  :on-click #(swap! state assoc-in
            [:rooms :kitchen :dish-washer :active?] true)}])
    ..
    ])
```

Listing 10: Kjøkken HTML

```
(defn stove-on []
  (when (and (:active? (:stove (:kitchen (:rooms @state))))
             (= :out (:current-location (:user @state))))
    (swap! state assoc :view :kitchen)))
```

Listing 11: Regel

4 Eksperimenter & Resultater

4.1 Evalueringsstrategier

For at en hypotese skal bevises gjennom den vitenskapelige metode kreves det logisk argumentasjon, empiriske bevis og at eksperimentet er reproduserbart. Alle hypotesene jeg har dannet utforskes i dette kapitellet med argumentasjon, men de empiriske bevisene vil mangle i noen tilfeller. Dette er rett og slett fordi det vil kreve mer tid og utførelse av brukertesting for å bevise disse hypotesene. Gestegjenkjennelsen vil evalueres av den prosentvise korrekte klassifiseringen. Den vil også evalueres etter hvor mange gester det lærte systemet kan skille mellom, mot hvor mange som kan eksplisitt programmeres. Dette gjelder hypotesene 1 og 3. Hypotesene 2 og 4 lener seg hovedsaklig på argumentasjon.

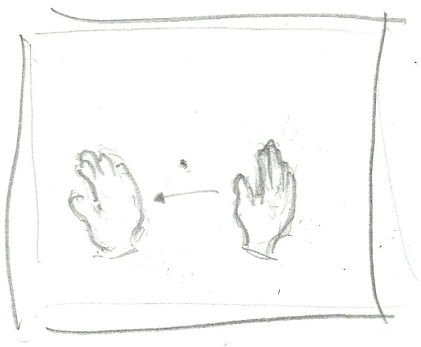
4.2 Eksperimentsutførelse

Gestegjenkjennelse gjennom fotodioder

Dersom man ønsker å tilby styringsmuligheter på en eller flere vegger i et hjem, kan enkle gestesensorer benyttes i stedet for et panel av knapper og dimmere. I det forrige kapitellet så vi at den aktuelle sensoren er på størrelse med et knappenålshode. Gode produktdesignere vil med dette utgangspunktet kunne kan lage et produkt som enten forsvinner inn i hjemmiljøet, eller et som synes tydelig, men er praktisk og estetisk. Sensoren merker at en hånd eller et annet objekt befinner seg foran den ved å sende ut et svakt infrarødt signal som reflekteres. Signalet detekteres dersom signalet er tilstrekkelig sterkt nok når det returnerer. Dette vil bare skje dersom objektet er opptil 20 cm unna sensoren. Gester forstås dermed kun når de utføres rett foran sensoren.

I motsetning til forståelse av gester gjennom kameraer og datasynsalgoritmer er dette altså en langt mindre påtrengende måte å interagere med brukerne. De kan være sikre på å verken være overvåket eller at personvernet deres på noen måte krenkes. Gestesensoren fungerer som en multifunksjonell knapp.

Figur 10 viser en typisk gest der brukeren sveiper hånda foran sensoren som befinner seg på veggen. Kanskje denne gesten betyr å skru av lyset i rommet sensoren befinner seg i. Eller kanskje gesten betyr å skru på radioen, låse opp en dør, lukke gardinene eller starte kaffemaskinen. Det er ingen begrensning på hva en enkelt gest aktiverer i form av funksjonalitet.



Figur 10: Gest: brukeren sveiper hånd mot venstre

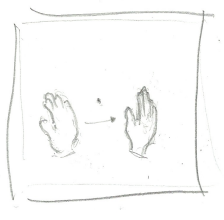
Når en gest utføres må enten rådataene fra sensoren eller en forståelse av dataene sendes til et system som har ansvaret for å styre hjemmet. Sensoren må være knyttet til en mikrokontroller eller tilsvarende som har ansvaret for å sende dataene videre. Dette kan enten være gjennom kabel eller trådløst. Det er mulig å programmere en mikrokontroller til å skille mellom sensordataene og forstå seks ulike gester[23]. Dette er i seg selv bra og betyr at en enkelt sensor kan fungere som en knapp med seks forskjellige innstillinger. Hvis man også utnytter at kombinasjonen av flere gester etter hverandre kan bety egne kommandoer er styringsmulighetene mange. Det finnes et alternativ til å programmere hva de ulike dataene skal tolkes som. Alternativet er å sende rådataene til en kraftigere maskin som kan benytte maskinlæring til å forstå potensielt enda flere ulike gester.

For å utføre eksperimentet benyttet jeg først Python-scriptet *process-data.py*, for å åpne tilkoblingen til den serielle porten mikrokontrolleren sender data til. Porten åpnes periodisk i noen sekunder og ber om at en gest utføres. Jeg gjennomførte 50 utførelser av hver av de 10 gestene (se figur 11), for totalt 500 treningseksempler. I illustrasjonene i figur 11 indikerer en enkelt pil et rolig sveip over sensoren. Tre piler indikerer en større hastighet. Dataene fra hver gest ble lagret i individuelle filer i *CSV*-format. Med dataene fra hver gest lagret i forskjellige filer, kan disse kombineres slik jeg ønsker og trene systemet til å skille mellom 2,3, eller opp til 10 gester.

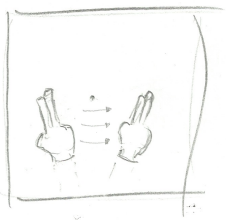
Maskinlæring handler om å la maskinen lære fra data. Dette kan enten være et forsøk på å finne ukjente sammenhenger i dataene den mates med eller det kan være å lære seg sammenhenger mellom treningseksempler og klasser. Det er dette sistnevnte scenariet jeg

[23] https://github.com/sparkfun/APDS-9960_RGB_and_Gesture_Sensor

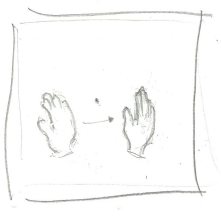
er interessert i. Jeg matet maskinen med data fra en gest og ga samtidig informasjon om at dataene maskinen akkurat fikk betyr "sveip til høyre". Dermed kan maskinen forsøke å danne en optimal kobling mellom dataene som kom inn da jeg sveipet til høyre og informerte om at disse dataene hørte til klassen "sveip til høyre". Med en tilstrekkelig mengde treningseksempler fra de ulike gestene skal maskinen kunne lære seg forskjellene mellom dem. Dermed vil den være i stand til å gjette riktig når en av de 10 gestene utføres senere. Jeg



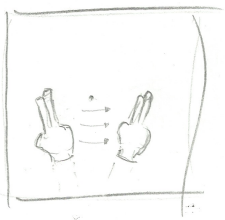
(a) Sveip til høyre.



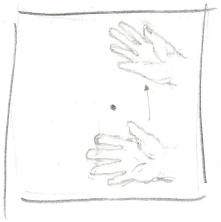
(b) Flikk til høyre.



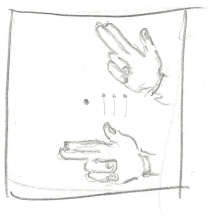
(c) Sveip til venstre.



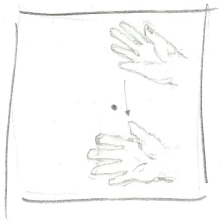
(d) Flikk til venstre.



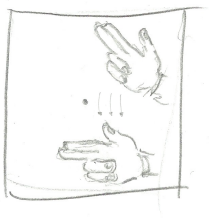
(e) Sveip opp.



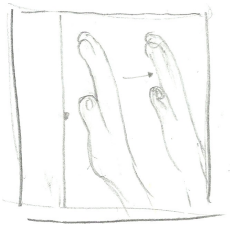
(f) Flikk opp.



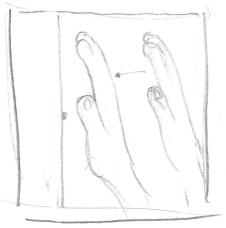
(g) Sveip ned.



(h) Flikk ned.



(i) Nær - fjern.



(j) Fjern - nær.

Figur 11: De 10 ulike gestene

benyttet Python-scriptet *learning.py* for å utføre maskinlæringen. De lagrede dataene ble lastet inn og de aktuelle klassene opprettet. Jeg splittet så dataene i et treningssett og et testsett. Splittingen var 75% til trening og 25% til testing. Dette er et viktig steg i prosessen for å ikke overspesialisere modellen til dataene. Dersom man trener algoritmen på hele datasettet og så tester på det samme settet, er det en sjanse for at modellen blir overtilpasset. Man ønsker å finne den underliggende sammenhengen i dataene. Man ønsker ikke å danne en modell som perfekt representerer treningsdataene, men som ikke vil håndtere nye data

godt. Hver modell ble trent og testet 100 ganger med et tilfeldig utgangspunkt hver gang. Sluttresultatet er et gjennomsnitt av disse 100 utførelsene, med et 95% konfidensintervall.

Multimodal interaksjon gjennom tale og gester

Mennesker i mellom kommuniserer i hovedsak med tale og det er derfor forsket mye på å få datamaskiner til å forstå og bruke denne naturlige kommunikasjonsformen. Det er liten tvil om at det er en stor drøm for mange å ha en kontinuerlig samtale med datamaskinen. I et smart hjem kunne det vært hendig å styre miljøet og enhetene, få hjelp til arbeid eller hobbyer og å gjøre oppslag etter matoppskrifter og værutsikter; alt gjennom en samtale med hjemmet.

La oss fokusere på å bruke tale til det folk bruker tale til; å gi kommandoer og å stille spørsmål. Tale har først om fremst sin plass som *kommunikasjonsprogramvare*. Tale kan også ha en begrenset nytte i forbindelse med læring; som å få faktasvar på tilstanden til hjemmet, eller for å lære om været for de neste dagene. Men for å bedrive dyp og god læring kreves det en aktiv utforskning. Vi lærer best ved å peke på ting, justere dem, holde dem i hendene og ved å ha muligheten til å bevege oss rundt dem. Tale virker upassende for denne typen aktiv utforskning. Når det gjelder programvare for å *skape* virker det også som om tale kommer til kort. Jeg kan forestille meg at det i framtiden blir laget verktøy der brukeren forteller datamaskinen hva som skal gjøres for å lage noe. Men arbeid som involverer en evolusjon av produktet, som kunst eller ingeniørvirksomhet, er vanskelig å beskrive med ord. Disse tingene skapes gradvis og vi manipulerer dem best med hendene våre. For meg fremstår tale hovedsaklig som en effektiv og naturlig interaksjonsform når det gjelder å gi kommandoer og å få informasjon, enten kommunikasjonen er med menneske eller maskin.

Dagens kommersielle løsninger, fra Google, Apple og andre, benytter seg av avanserte systemer for å forstå kontinuerlig tale. Med optimaliserte algoritmer og tilstrekkelig datakraft tilbyr disse aktørene talegjenkjenning med en svært høy grad av korrekthet. Men for å benytte tjenestene må taledataene som oppfattes av brukerens mikrofon, sendes til kraftige maskiner på andre siden av kloden. Der prosesseres dataene, talen forstås og resultatet returneres. Det kan være vanskelig for disse systemene å garantere en god responstid. Dersom

det tar flere sekunder før et resultat returneres vil nytteverdien til talegjenkjenning raskt minke. Umiddelbar respons vil være langt mer attraktivt. Denne tilnærmingen til talegjenkjenning har også problemer rundt bevarelse av personvernet. I et hvert tilfelle hvor data forsvinner ut av brukerens kontroll kan det være problemer. Det kan tenkes at taledata til tider inneholder sensitiv informasjon, som kalenderinformasjon, søkeord, eller tilstanden til hjemmet. Dersom informasjon om hjemmets alarmsystem, tilstedeværelse av brukere eller ulåste dører forsvinner ut av hjemmet, er dette en direkte sikkerhetsrisiko. En kjeltring, ute etter gjenstander eller identiteter, kan sitte på utsiden av hjemmet og få tilgang til sensitive data om hjemmet og dets beboere.

Samsung tilbyr både tale- og gestekontroll på sine nyere Smart-TV-er. En nærmere kikk på deres *global privacy policy* skapte i år oppstyr i media. Samsung tilsier seg retten til å lagre tale- og video-data for å forbedre tjenesten, samt å gi den videre til visse tredjeparter[24]. Brukere har naturligvis vist misnøye med at alt de sier i stua kan bli tatt opp og sendt til tredjeparter. Samsung har advart brukerene om å diskutere personlig informasjon foran disse Smart-TV-ene. Personvernsbevisste brukere må altså nå passe på hva som blir sagt i sin egen stue!

Basert på diskusjonen så langt kan det se ut som talegjenkjenningsprosessen bør holdes lokal, og at man dermed unngår både treg responstid og mulig krenking av personvern. Er dette mulig i dag? Moderne telefoner, tv-er og andre enheter blir stadig kraftigere, men de er langt unna supercomputerne som for eksempel kjører Google's talegjenkjenning.

Egenskapene jeg omtalte i kapittel 3.4.2 gjør Pocketsphinx til et interessant valg som kommunikasjonsprogramvare for smarte hjem. Problemdomenet vårt tilsier at vi ønsker et system som gjenkjenner et titalls kommandoer i reelltid. Det virker derfor ikke som det er nødvendig å knytte forståelsen opp mot internettet og sende taledata ut i verden. Vi kan unngå problemene med personvern, som Samsung og andre styrer med, ved å holde alle data lokalt. Ved å benytte et begrenset vokabular ofrer vi den kontinuerlige og mer naturlig talen. Til gjengjeld får vi en garantert bevarelse av personvernet. Dette er et steg bakover i utviklingen teknologimessig, men framstår som det beste alternativet i smarte hjem.

[24] https://www.samsung.com/uk/info/privacy-SmartTV.html?CID=AFL-hq-mul-0813-11000170

I kapittel 3.4.2 nevnte jeg hvordan de fleste kommersielle produktene håndtere problemet med kontinuerlig lytting ved å benytte et aktiveringsord. Det finnes andre løsninger på dette problemet. Et avansert, men elegant, alternativ er å benytte stemmegjenkjenning. Her forstår systemet hvem som taler basert på talerens unike vokaltrakt. Systemet kan dermed programmeres til å kun godta kommandoer fra gjenkjente og klarerte brukere. En enklere løsning er å tilby talegjenkjenning i begrensede områder av hjemmet. For eksempel er det vanlig å tilbringe tiden alene på badet, så her kunne det vært aktuelt å la systemet lytte kontinuerlig da eventuell tale sannsynligvis vil være kommandoer til hjemmet. Jeg foreslår i dette eksperimentet å tilby kommandoutførelse gjennom tale kun dersom det også følger med en gest. Dette betyr at brukeren må stå foran enheten for å styre hjemmet. Enten kan en gest aktivere lyttingen, eller hjemmet kan lytte kontinuerlig, men kommandoer blir kun aktivert dersom en gest også utføres.

```
BATHROOM        B AE TH R UW M
BEDROOM B EH D R UW M
BLINDS  B L AY N D Z
CAR     K AA R
CLOSE   K L OW S
CLOSE(2)        K L OW Z
COFFEE  K AA F IY
COFFEE(2)       K AO F IY
CURTAINS        K ER T AH N Z
DECREASE        D IH K R IY S
DECREASE(2)     D IY K R IY S
DISHWASHER      D IH SH W AA SH ER
DOOR    D AO R
DOWN    D AW N
FRIDGE  F R IH JH
FRONT   F R AH N T
GARAGE  G ER AA ZH
HALL    HH AO L
INCREASE        IH N K R IY S
KITCHEN K IH CH AH N
LAUNDRY L AO N D R IY
LIGHTS  L AY T S
LIVINGROOM      L IH V IH NG R UW M
LOCK    L AA K
MACHINE M AH SH IY N
OFF     AO F
ON      AA N
ON(2)   AO N
OPEN    OW P AH N
OVEN    AH V AH N
PAUSE   P AO Z
PORT    P AO R T
POWER   P AW ER
RADIO   R EY D IY OW
SPEAKER S P IY K ER
START   S T AA R T
STOP    S T AA P
STOVE   S T OW V
TEMPERATURE     T EH M P R AH CH ER
TEMPERATURE(2)  T EH M P ER AH CH ER
TV      T IY V IY
TV(2)   T EH L AH V IH ZH AH N
UNLOCK  AH N L AA K
UP      AH P
VOLUME  V AA L Y UW M
WASHING W AA SH IH NG
```

Figur 12: Vokabular

Pocketsphinx trenger et vokabular slik at den kan knytte taledata til ord og setninger. Man skriver en liste med setninger eller enkeltord man ønsker at systemet skal forstå. Deretter kan man benytte et språkmodell-verktøy til å få laget en språkmodell og et vokabular. Jeg laget en ordliste med 38 elementer. Jeg valgte først å ta med substantiver; steder og ting i hjemmet, som ”kitchen”, ”bedroom”, ”dishwasher” og ”radio”. Det er også andre substantiver som beskriver ting eller egenskaper ved mange av rommene eller apparatene: ”lights”, ”temperature”, ”blinds”, ”curtains”, ”volume” og ”power”. Til sist er det med verb og preposisjoner, som ”on”, ”up”, ”start”, ”increase” og ”lock”. Med et slikt vokabular kan brukeren potensielt styre det smarte hjemmet med en stor grad av kontroll.

Figur 13: Multimodal input visualisert

Figur 12 viser det resulterende vokabularet etter at verktøyet har prosessert en liste med ord. Vi ser at verktøyet har forstått ordene som engelsk og har tilført data om engelsk uttale til listen. I noen tilfeller, som for "TV", er det flere alternative uttalser. Jeg skrev inn "TV" i ordlisten, og verktøyet har laget en forståelse for både "TV" og "television".

Jeg har allerede pekt på problemene med å kontinuerlig lytte etter tale. Dersom systemet er slik at tale og gester må kombineres for effekt, kan tale benyttes for å peke ut substantivet, eller hva som skal opereres, og gester kan brukes for å utøve verbet eller preposisjonen. For eksempel kan brukeren si "kitchen lights" og samtidig utføre et sveip mot høyre, for å skru på kjøkkenlyset. Eller brukeren kan heve persiennene ved å si "blinds" og sveipe hånda oppover. Figur 13 er en illustrasjon der input fra både gester og tale tegnes til det samme lerretet med noe forskjellige skriftstørrelse og farger. Denne enkle grafikken demonstrerer at dataene fra begge inputkilder er likestilt og at ingen data går tapt, selv med input fra flere simultane kilder. Dette er en form for *late/decision-level fusion*, der de to inputkildene forstås hver for seg og resultatene kombineres og vises grafisk.

Kombinasjoner

I det tredje eksperimentet definerte jeg 42 abstrakte gester, skapte data fra disse gestene og benyttet maskinlæring for å skille mellom dem.Deretter ble andre data fra sensorene forsøkt utnyttet for å skape en verdifull applikasjon i hjemmet. Et stort vokabular av abstrakte gester er i praksis et tegnspråk. Derfor burde alle retningslinjer for tale også gjelde i dette tilfellet. Teknologien fungerer best som kommunikasjonsprogramvare og jeg spår begrensninger innen de to andre kategoriene.

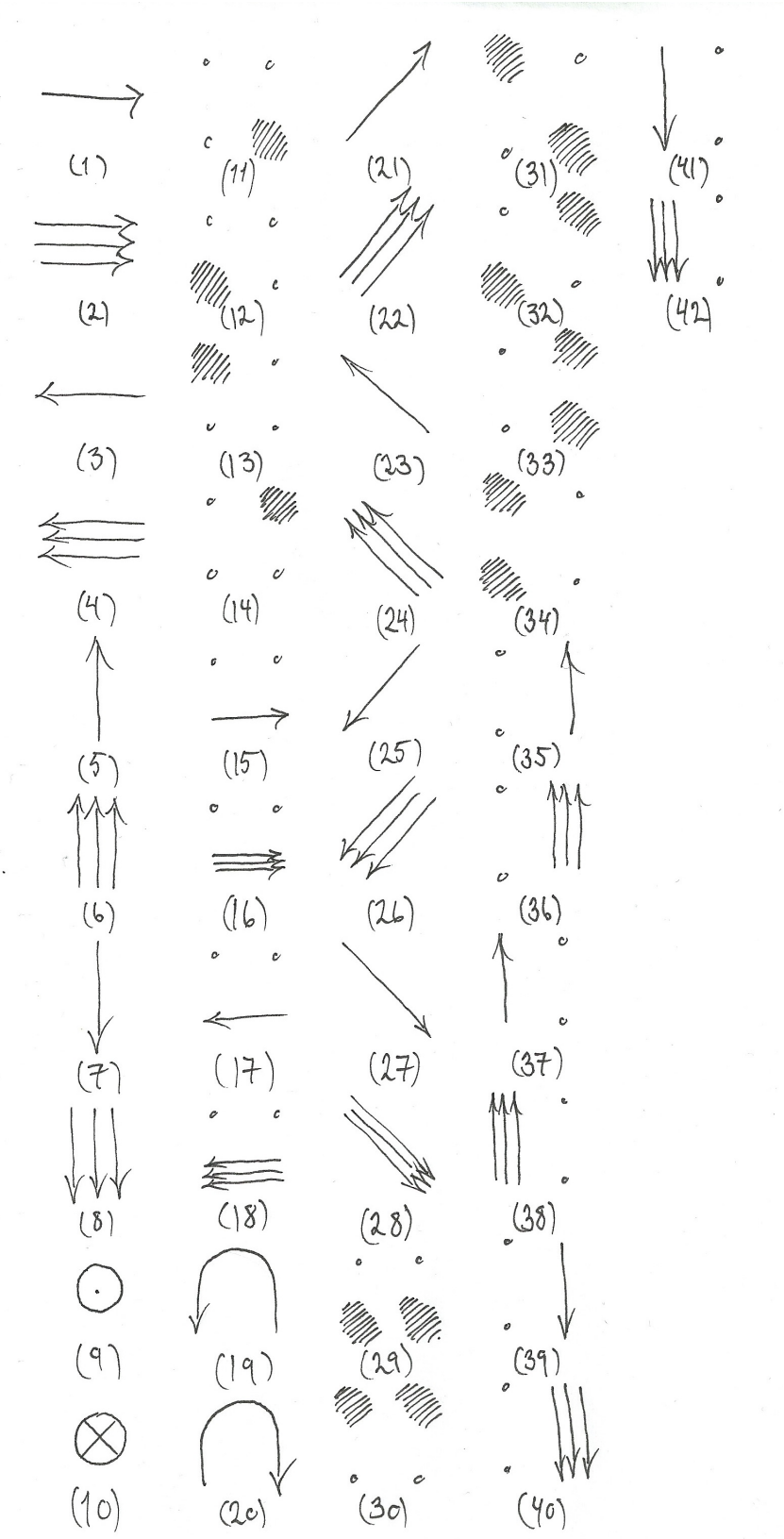
Figur 14 er en illustrasjon av de 42 gestene. De 10 første er de samme som i eksperiment 1. En pil indikerer et sveip i pilens retning med normal hastighet. Tre piler indikerer en høyere hastighet. Gestene 9 og 10 er gester der hånda starter enten nære eller fjernt fra sensoren, og føres i den motsatte retningen. De nye gestene er gester som nr 11; her er de fire sensorene tegnet opp, og en av dem er tildekt av en hånd. Jeg utførte denne delen av eksperimentet på samme vis som eksperiment 1. Dataene ble først produsert for hånd og deretter ble modeller trent med maskinlæring. Da det her var snakk om 42 ulike gester tok jeg meg bare tid til å lagre 10 treningseksempler av hver gest.

Den andre delen av hypotesen innebar å utforske de andre egenskapene sensoren har.Jeg kom opp med følgende: benytt nærhetssensoren som en reguleringsmekanisme for ikke-binære innstillinger. Figur 15 viser denne interessante bruken av nærhetssensoren, kombinert med tale. Nærhetssensoren gir data om hvor nære et objekt befinner seg. Brukeren kan uttale hvilket apparat hun ønsker å styre og deretter dynamisk styre innstillingsnivået ved å variere avstanden mellom hånda og sensoren. Eksempelbruk kan være å regulere volumet fra høytalere, temperatur eller lysstyrke, som illustrert i figur 15.

Kontekstdrevet brukergrensesnitt

Som regel når en person bruker programvare er det ikke for å skape noe, men for å lese, observere, utforske og lære. Folk er ute etter å ordne egne tanker. Datamaskinen er et medium for å spørre spørsmål, gjøre sammenligninger og dra konklusjoner. Det aller meste av programvare er altså *informasjonsprogramvare*.

Informasjonsprogramvare er et medium for visuell kommunikasjon. Utseende og hvordan



Figur 14: 42 gester

(a) (b) (c) (d) (e)

Figur 15: Dimmeeffekt

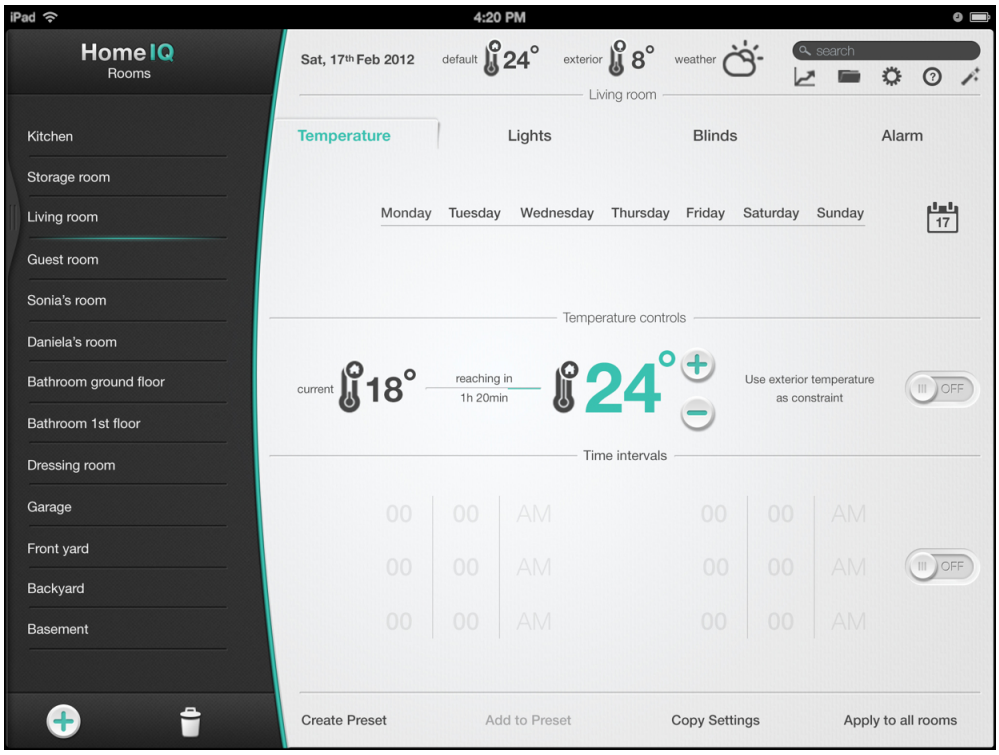
informasjon presenteres bør være i fokus. Hva er den relevante informasjonen? Hva vil brukeren vite? Hvordan kan data vises på den mest effektiv måten? Kan teknikker fra grafisk design anvendes for å lede brukerens blikk mot løsningen?

Delen av grafisk design mest relevant til programvare er det Tufte (2001) kaller informasjonsdesign; bruken av bilder for å uttrykke informasjon av interesse for brukeren. En god grafisk designer posisjonerer informasjonen slik at leseren kan få svar på spørsmål, gjøre sammenligninger og trekke slutninger. Når programvareutvikleren definerer en visuell representasjon i programmet sitt, utføres en form for grafisk design, nettopp fordi det handler om å definere og posisjonere bildene brukeren skal forstå. Figur 16 viser et brukergrensesnitt jeg fant på en nettside der folk viser fram prosjekter innen interaksjonsdesign[25]. Jeg har sett flere brukergrensesnitt for smarte hjem og dette er det beste jeg har funnet. Grensesnittet er pent å se på og det ser ut til å tilby mange innstillinger. Men hvorfor må brukeren trykke på alle knappene for å lære om hjemmet? Dataene fra et smart hjem er svært rike, med sensorer, apparater og aktuatorer. Hvorfor viser vi ikke mer av den tilgjengelige informasjonen?

Her er noen spørsmål brukeren kan ha til det smarte hjemmet:

- Hvor lenge er det til vaskeprogrammet er ferdig?
- Står lyset på i garasjen?
- Hva er temperaturen i stua nå?
- Trenger jeg å handle inn matvarer?

[25] Design av Calin Giubega http://www.studentshow.com/gallery/HomeIQ-Smart-Home-System/7740705, gjenbrukt under creative commons-lisens https://creativecommons.org/licenses/by-nc-nd/3.0/

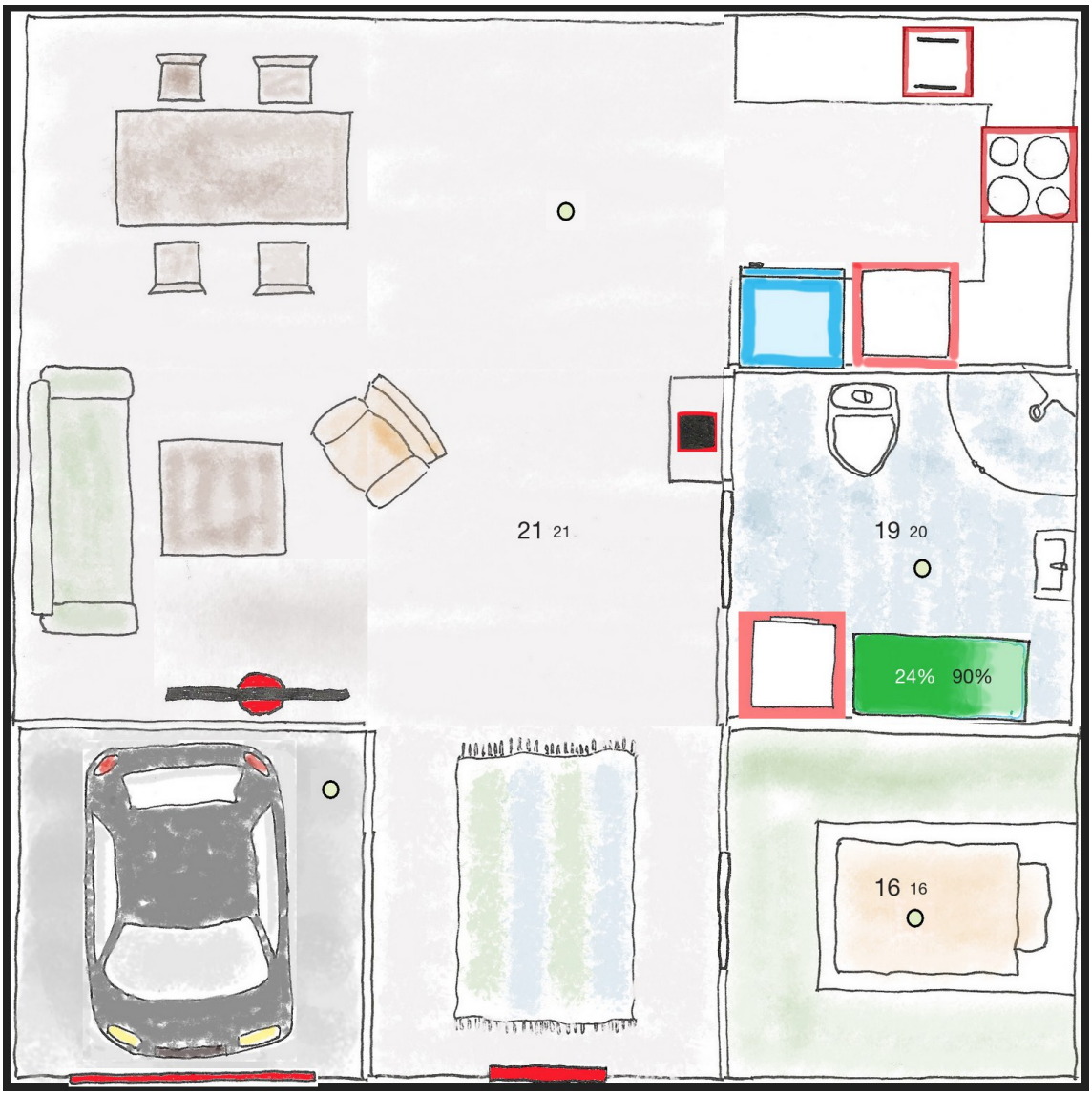


Figur 16: Brukergrensesnitt med fokus på interaksjon

- Er ytterdøra og garasjeporten låst for natta?
- Står det på unødvendig lys og varme i rom vi sjeldent bruker?
- Hvor mye strøm brukte vi forrige måned?

Hvordan kan disse spørsmålene best besvares? I grensesnittet i figur 16 må brukeren ha en interaksjon med systemet; han må lete seg gjennom estetiske menyer for å finne informasjonen. Tufte (2001) har regler for grafisk design og den første er "framfor alt, vis dataene". Vi bør designe mot å kunne svare på spørsmål ved å vise dataene. En godt designet informasjonsgrafikk kan nærmest nøde brukeren til å stille og svare spørsmål, gjøre sammenligninger og dra slutninger. Dette oppnås ved å utnytte øyets egenskaper til å håndtere store mengder visuell data og gjenkjenne mønstre og sammenhenger. Dette henger sammen med Schneidermans argument i **??**. Men i stedet for å tilby alle mulige knapper, kan vi i stedet bare vise dataene!

Tilsvarende viktig som hva slags data som vises, er hvor dataene vises. Grafikkelementer kan plasseres på bestemte steder for å utnytte vår romforståelse. Desverre er eksisterende



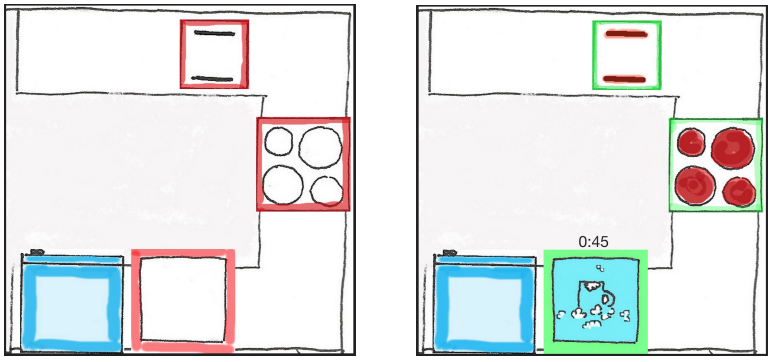
Figur 17: Smart hjem som grafisk design

programvare for smarte hjem presentert for å maksimere estetikk, ikke for å framheve sammenhenger i dataene. Grensesnittet i figur 16 er pent å se på, men setter behagelig interaksjon foran presentasjon av dataene. Vi kjenner hjemmene våre som fysiske steder. Vegger, dører, møbler og apparater har geografiske posisjoner i forhold til hverandre. Mennesket fant for mange å siden en effektiv abstraksjon for å representere geografisk data, ved *kartet*. Å vise informasjon om hjemmet som et kart utnytter vår eksisterende kunnskap om de romlige relasjonene i hjemmet. Figur 17 presenterer hjemmet som et to-dimensjonalt kart[26]. Hva slags

[26]Programvaren kan testes på https://thesmarthome.herokuapp.com/

funksjonalitet skal dette grensesnittet ha? Hva er en rimelig reaksjon fra denne type programvare, når brukeren trykker på ulike apparater? Å klikke på en lysbryter (små, blekt gule sirkler) bør naturligvis skru av eller på lyset. Å klikke på kjøleskapet eller vaskemaskinen bør enten skru av og på eller gi mer informasjon om disse apparatene. Dersom en bruker trykker på en knapp og ingenting skjer, er det vanskelig å forstå om handlingen utførte noe som helst. Brukeren får ikke muligheten til å evaluere en respons og la den lede til en ny handling. Interaksjon med brukergrensesnittet skaper kontekst. Enhver interaksjon bør gi en merkbar forandring i grafikken. Å gi direkte tilbakemelding reduserer mengden manipulasjon brukeren må utføre for å nå en tilstand der den ønskede informasjonen er tilgjengelig. I stedet for å vite navnet på en gjenstand eller et rom kan brukeren peke på kartet og si "der!". Dette vil være svært viktig i en situasjon i framtiden der hjemmet har et stort antall apparater som er koblet til nettet. *Feedback-loopen* med denne løsningen er svært tett; dersom brukeren pekte på et element som kan interageres med, oppdateres grensesnittet umiddelbart. Brukeren ser alltid informasjonen om hjemmet. Dersom det som vises ikke er konfigurasjonen brukeren ønsker, kan det forandres på stedet. Det er ingen *OK*, *submit* eller *apply*-knapp. Kartet viser alltid den nåværende tilstanden til hjemmet.

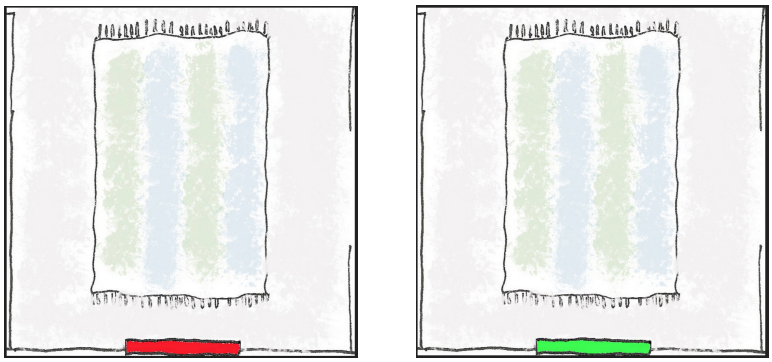
Det er noen forsøk på å bruke farger symbolsk i grafikken. Elementer med klare rødfarger



(a) (b)

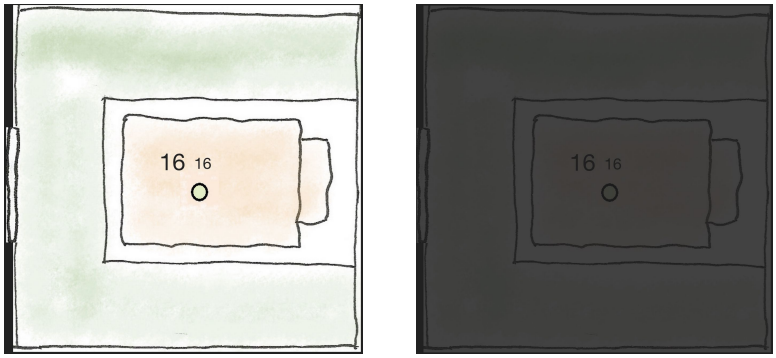
Figur 18: Apparater skrudd av og på

er avslått eller låst. Elementer med klare grønnfarger er på eller åpne. Figur 18 og 19 viser forskjellene på elementer som er avslått/låst og som er på/åpne. Kjøleskapet er i en klar farge for å indikere at den kan interageres med. Ellers er elementene som utgjør bakgrunnen i grafikken i grå, utvaskede eller sjenerte farger. Noen av elementene i bakgrunnsgrafikken er



(a) (b)

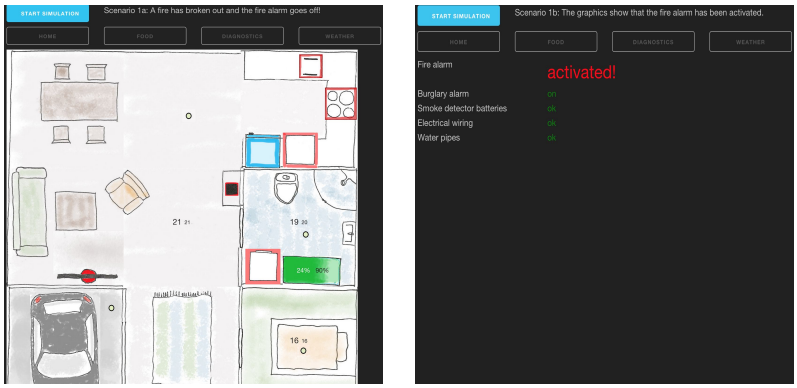
Figur 19: Gang med dør låst og ulåst



(a) (b)

Figur 20: Soverom med lyset slått på og av

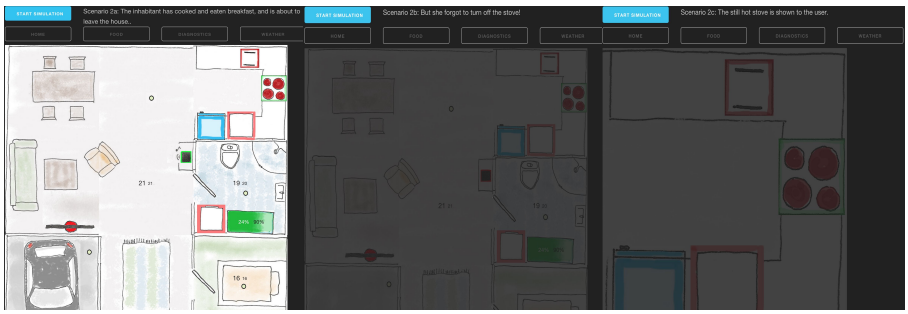
også dynamiske og kan forandres av data om tilstanden til hjemmet. Dette gjelder dørene til bad og soverom, samt skittentøyskurven. Disse er dynamiske, men kan ikke interageres med. Dette bør gi mening ettersom det ikke er mulig å fjernstyre dørene i hjemmet. Det er også benyttet enkle animasjoner for å vise apparater som er aktive. Disse involverer forandring i lys fra tv, strømmende noter fra radio, og vaske- og oppvaskmaskiner med bevegelse og forandrene såpebobler. Figur 20 viser hvordan soverommet ser ut med lyset slått på/av. Lyset manipuleres ved å trykke på den runde, blekt gule sirkelen i midten av rommet. Dersom systemet kan forstå konteksten i hjemmet kan irrelevant data fjernes og grafikk som er generes etter behov i øyeblikket. Tradisjonelt er konteksten definert av brukerinteraksjon. Søkeord og valg i menyer gir programvaren informasjon om hva som er viktig og gir den muligheten til å presentere relevante resultater. Kontekstinformasjon kan komme fra sensorinformasjon fra det fysiske miljøet, historie over tidligere valg og gjennom interaksjon med brukeren. Det



(a) (b)

Figur 21: Scenario 1

smarte hjemmet vil være fylt til randen av sensorer og informasjon fra det fysiske miljøet, så dette kan regnes som den viktigste kontekstinformasjonen. Annen informasjon, som tid og hjemmets posisjon, gjør det enkelt å hente informasjon fra eksterne kilder, som å spørre *API*-er etter værutsiktene eller brukerens kalenderinformasjon. Systemet kan også bruke historie

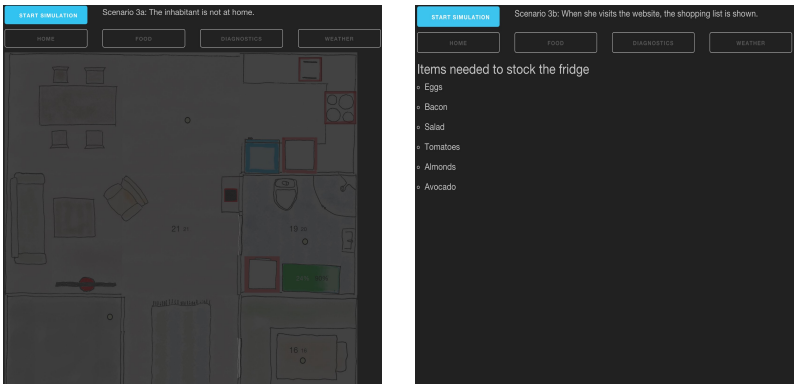


(a) (b) (c)

Figur 22: Scenario 2

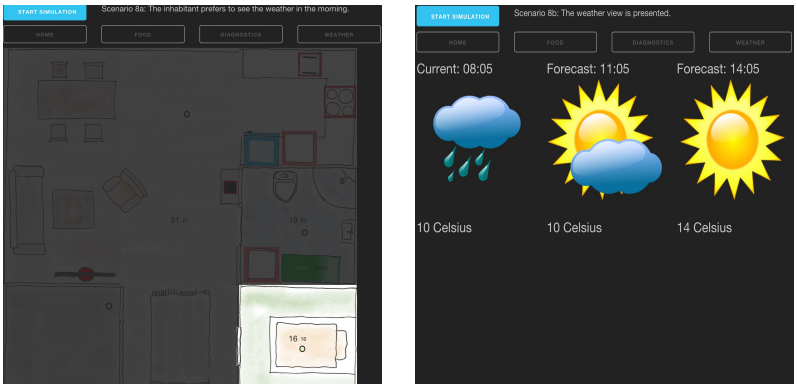
til å forstå kontekst. Den nåværende konteksten, eller i det minste en god tilnærming, kan bli gjettet basert på en historie av tidligere miljøer og interaksjoner. Å gjette basert på den siste kjente verdien, er den enkleste formen for gjetning. Systemet gjetter den nåværende konteksten til å være den samme som den forrige. Dette er rimelig i mange situasjoner der konteksten forandrer seg lite over et kort tidsrom. For eksempel, hvis brukeren i går foretrakk å se på en grafisk representasjon av hele huset, er den samme brukeren sannsynligvis interessert i å se det samme den neste dagen. Det ville ikke vært rimelig å neste dag starte med å vise informasjon om energiforbruket i huset. Hvis lite, eller ingenting er forandrert bør

programvaren vise den samme informasjonen som var foretrukket sist. Dersom systemet



(a) (b)

Figur 23: Scenario 3



(a) (b)

Figur 24: Scenario 8

kan forstå så mye som mulig fra miljøet og fra historie bør den i det minste klare å produsere et rimelig utgangspunkt for å gi brukeren det hun er interessert i. Mesteparten av brukerens interaksjon blir derfra å korrigere eller bekrefte programmets gjetninger.

For å demonstrere hvordan grafikken forandrer seg dynamisk basert på dataene fra hjemmet implementerte jeg en simulering som går gjennom et utvalg scenarier. Disse scenariene er definert av regler. Reglene sier hva som skal vises dersom visse data forekommer. Figur 21 viser hva som skjer dersom brannalarmen aktiveres. Dersom brukeren besøker nettstedet blir hun straks møtt med tydelig informasjon om at brannalarmen har blitt aktivert. Figur 22 viser hvordan hjemmet presenteres dersom brukeren forlater hjemmet uten å skru av stekeplata. Den potensielt farlige situasjonen settes i fokus og lar brukeren enkelt skru

av plata dersom det er ønskelig. I figur 23 er hjemmet i en nøytral tilstand og brukeren er ikke i hjemmet. Han har laget en regel som sier at kjøleskapets handleliste skal vises i denne situasjonen. Dette er den informasjonen om hjemmet denne brukeren som oftest er interessert i når han er ute og går. Til sist, i figur 24 viser systemet været om morgenen.

4.3 Resultater

Gestegjenkjennelse gjennom fotodioder

Resultatet av dette eksperimentet vises i følgende tabell. Tabellen viser graden av suksess de lineære klassifiseringsalgoritmene oppnådde. Modellene er trent og testet 100 ganger med

% korrekt klassifisering	Algoritme	Antall gester	Treningseksempler/gest
96.22 +- 0.34	SVM v/libsvm	10	50
96.02 +- 0.35	SVM v/liblinear	10	50
94.58 +- 0.34	Logistisk regresjon	10	50

Tabell 2: Klassifisering av 10 gester

tilfeldige utgangspunkt og resultatene er oppgitt med 95% konfidensintervall.

Multimodal interaksjon gjennom tale og gester

Dette eksperimentets resultat ble en argumentasjon for begrenset tale i hjemmet. Det ble også foreslått en måte å håndtere multimodaliteten på, med tråder, en felles kø og *late fusion*.

Kombinasjoner

% korrekt klassifisering	Algoritme	Antall gester	Treningseksempler/gest
94,04 +- 0.68	SVM m/liblinear	10	10
93,24 +- 0.75	Logistisk regresjon	10	10
92,52 +- 0.75	SVM m/libsvm	10	10

Tabell 3: Klassifisering av de samme 10 gestene fra eksperiment 1

Modellene er trent og testet 100 ganger med tilfeldige utgangspunkt. Med kun 10 tren-

% Korrekt klassifisering	Algoritme	Antall gester	Treningseksempler/gest
93,68 +- 0.55	SVM m/libsvm	42	10
92,74 +- 0.49	Logistisk regresjon	42	10
92,54 +- 0.52	SVM m/liblinear	42	10

Tabell 4: Klassifisering av 42 gester

ingseksempler av hver av de 42 gestene oppnås en suksessrate på 93.68%. Modellene er trent og testet 100 ganger med tilfeldige utgangspunkt. Resultatene er oppgitt med 95% konfidensintervall.

Eksperimentet har i tillegg vist hvordan dimming kan implementeres med en nærhetssensor.

Kontekstdrevet brukergrensesnitt

Dette eksperimentet resulterte i en implementasjon av et grafisk brukergrensesnitt, designet delvis etter prinsipper fra grafisk design. Utviklingen ble en utforskning av et dynamiske brukergrensesnitt, drevet av kontekstdata fra hjemmet, framfor interaksjon med brukeren. Eksperimentet benyttet prioriterte regler for å avgjøre hvordan dataene burde presenteres.

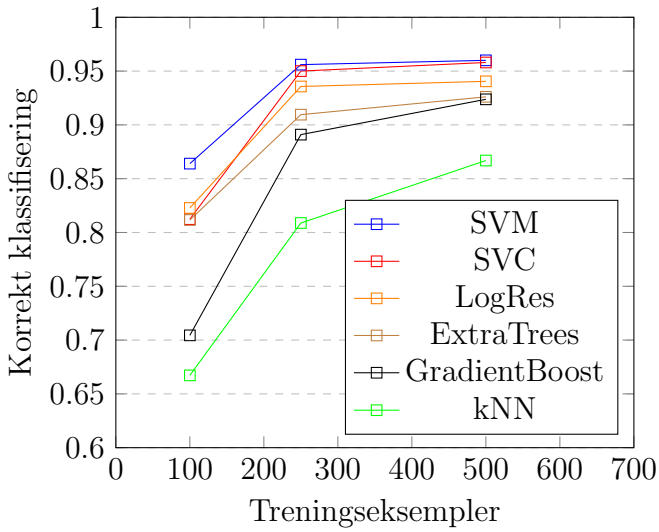
5 Diskusjon & Konklusjon

5.1 Diskusjon

Gestegjenkjennelse gjennom fotodioder

Eksperimentet oppsummert:

- Jeg argumenterte for at en gestesensor i form av enkle fotodioder er et tilstrekkelig medium for enkel brukerinteraksjon.
- Jeg viste at maskinlæring kan benyttes for å lære et system å forstå enkle gester og viste at dette er et alternativ til å eksplisitt programmere forståelse.
- Jeg viste at det holder med et titalls treningseksempler fra hver gest for å oppnå gode resultater med lineære modeller, og at det med 50 eksempler oppnås en suksessrate på 96%.



Figur 25: Resultatsutvikling for et utvalg algoritmer

Både logistisk regresjon og støttevektormaskin ga svært lovende resultater, som vist i tabell 2. 96% er en meget høy suksessrate og viser hvor godt enkle, lineære modeller kan skille på denne typen data. Det ble interessant å deretter spørre om dette resultatet kunne blitt enda høyere. Jeg ønsket ikke å bruke mer tid på å lage flere treningseksempler, så i stedet utførte

jeg treningen på nytt med færre treningseksempler, i håp om å kunne se en utviklingstrend. Det samme eksperimentet ble utført med en femtedel og halvparten av dataene for å danne et bilde av sammenhengen mellom forbedring i suksessrate og antall treningseksempler. Figur 25 viser resultatene for 100, 250 og 500 treningseksempler. Ettersom biblioteket jeg benyttet for å implementere algoritmene tilbyr en rekke andre algoritmer forsøkte jeg noen andre tilnærminger enn lineære modeller og inkluderte dem også.

I figur 25 kan vi se at SVM-ene er i nærheten av 95% allerede etter 250 treningseksempler og at de kun øker minimalt med 250 ekstra tilfeller. Dette tyder på at det trengs et stort antall ekstra treningseksempler for at algoritmene skal krype betydelig nærmere 100%. De mer kompliserte algoritmene *ExtraTrees* og *GradientBoost* benytter seg av flere algoritmer under panelet og kombinerer disse. Det kan se ut som om spesielt GradientBoost kan fortsette å øke suksessraten betydelig med mer trening, men det er tvilsomt om den noen gang passerer SVM. Til sist nevnes k-nærmeste nabo (kNN), som begynner svakest av de utprøvde algoritmene, men fremdeles har en sterk vekst mellom 250 og 500 tilfeller. Det hadde vært interessant å se utviklingen videre for denne enkle algoritmen.

Dette prosjektet har argumentert for at gester kan være en aktuell interaksjonsform i hjemmet, spesielt som en erstatning til store knappepaneler og desentralisert styringskontroll. Det har også vist at maskinlæring kan benyttes for å gi enkle sensorer en svært god forståelse av gester. Det er dermed vist at det ikke er nødvendig med kameraer for å benytte gester. Med en suksessrate på over 96% i klassifiseringen av 10 ulike gester er denne teknikken svært interessant. Og med en suksessrate på over 85% allerede etter kun 10 treningseksempler på hver gest, kan man forestille seg at brukere selv kan sette av 10 minutter til å trene et helt nytt og utrent system til å forstå sine egne gester. I et produkt kunne det vært aktuelt å tilby online-læring gjennom systemets levetid. Man kan med andre ord la systemet lære etter hvert som brukeren benytter systemet. Dette vil nødvendigvis kreve at brukeren har en mulighet til å gi tilbakemelding på når systemet gjettet riktig gest og når det gjettet feil. Jeg har vist at en enkel sensor, gjennom maskinlæring, kan benyttes praktisk som en multifunksjonell bryter med flere tilstander enn hva man kan programmere for hånd.

Multimodal interaksjon gjennom tale og gester

Eksperimentet oppsummert:

- Jeg argumenterte for at tale og gester fungerer best som kommunikasjonsprogramvare.
- Jeg argumenterte for at kombinasjonen av enkle gester og talekommandoer er en effektiv måte å interagere med det smarte hjemmet på.
- Jeg viste at talegjenkjenning over en begrenset mengde ord dekker funksjonaliteten vi ønsker å tilby gjennom tale, uten problemer med personvern.
- Jeg implementerte et system som håndterer multimodal inputdata fra gestesensor og mikrofon. Denne ble benyttet for å simulerer bruken i et smart hjem ved å vise en dynamisk grafisk representasjon av inputdataene.

Hypotesen kan sies å være godt utfordret dersom den multimodale interaksjonen kan vises å være tilstrekkelig til å styre hjemmet, at den er naturlig og at den er effektiv. Disse er vanskelige å måle empirisk. Eksperimentet har argumentert for at den begrensede talegjenkjenningen ikke bare er tilstrekkelig, men er en bedre tilnærming enn kontinuerlig tale, med dagens begrensninger i teknologi. Med definisjonene for naturlig og effektiv jeg ga i kapittel 1 trengs det eksperimenter med brukere for å besvare hypotesen. For å avgjøre om interaksjonen er instinktiv, forståelig, enkel i bruk, praktisk og hensiktsmessig må et bredt utvalg brukere observeres mens de tar i bruk systemet. Jeg definerte også ordene slik at naturlig interaksjon må gi rimelige resultater og effektiv interaksjon må gi resultater raskt. I kontekst av det smarte hjemmet kan disse kravene være vanskelig å oppfylle. Dersom du sier til hjemmet at lyset skal skrus av i rommet brukeren befinner seg i, kan resultatet evalueres umiddelbart. Men dersom brukeren bruker systemet for å låse ytterdøra, senke temperaturen eller slå av lyset i et annet rom er det en utfordring å for dette systemet å gi feedback raskt, og å gi feedback som forteller brukeren om resultatet var rimelig.

Kombinasjoner

Eksperimentet oppsummert:

- Jeg argumenterte for at et stort vokabular av abstrakte gester i praksis er et tegnspråk og at retningslinjer som gjelder for taledrevne systemer også gjelder for slike gestedrevne systemer.

- Jeg viste at de 10 gestene fra eksperiment 1 danner langt bedre data med fire sensorer, og at maskinlæring er enda mer effektivt.

- Jeg viste at systemet kan trenes til å skille mellom 42 ulike gester med 93% suksessrate, med kun 10 treningseksempler av hver gest.

- Jeg argumenterte for at sensorenes nærhetsdeteksjon kan brukes i forbindelse med smarte hjem, for eksempel for å dimme lys og regulere lydvolum.

Resultatet fra maskinlæringen bekrefter hypotesen. Om de 42 gestene er naturlige er uvisst og krever brukertesting. Gestene er enkle og bør derfor også være forståelige. Om gestene gir rimelige resultater avhenger av hvordan de anvedes. Jeg vil si at en sveipende gest er like naturlig for å skru av/på et lys, som en tradisjonell knapp er. Det samme gjelder for å regulere nivåer med avstand, mot å skru på et hjul. Andre gester kan kanskje anses som mer abstrakte og mindre naturlige.

I eksperiment 1 økte resultatene fra 86%, 82% og 81% med 10 treningseksempler av hver gest, til 96%, 96% og 95% med 50 treningseksempler av hver gest. Dette er en økning på 11.6%, 17.1% og 17.3%. Dersom vi kan anta en liknende forbedring i 50 treningseksempler av hver av de 42 gestene kan vi forvente resultater som kryper svært nære 100%.

Kontekstdrevet brukergrensesnitt

Eksperimentet oppsummert:

- Jeg argumenterte for at et grafisk brukergrensesnitt for smarte hjem er informasjonsprogramvare. Brukerene er hovedsaklig interessert i å lære om hjemmet. Informasjonen bør derfor presenteres på en optimal måte, for å oppnå dette målet.

- Jeg implementerte et brukergrensesnitt som viser hjemmets tilstand og dynamisk tilpasser

seg ny informasjon. Grensesnittets utseende forandret seg basert på regler og prioriteringer.

Brukergrensesnittet jeg har utviklet og argumentert for forsøker å maksimere utnyttelsen av de rike kontekstdataene et smart hjem tilbyr. Ved å forstå konteksten rundt dataene gjennom regler, kan programvaren hjelpe brukeren med å finne, og lære av, informasjonen som er av interesse. Disse egenskapene gjør brukergrensesnittet til et intelligent brukergrensesnitt.

5.2 Videre arbeid

Gestegjenkjennelse gjennom fotodioder

Tre viktige valg ble tatt i dette eksperimentet: hvilke klassifiseringslagoritmer som ble utforsket, antallet treningseksempler og antallet datapunkter i treningseksemplene. Alle disse dimensjonene ville vært interessante å utforske med andre valg. Vil andre algoritmer fungere bedre dersom antallet treningseksempler øker? Teknikker som dyp læring med nevrale nett kan tilnærme mer avanserte funksjoner med tilstrekkelig trening. Kunne resultatene vært bedre med større datavektorer? Med gestene som skaper mange datapunkter blir datapunktene omgjort til gjennomsnittsresultater og vi mister en del presisjon. Treningseksempler med mer data vil kreve flere ressurser fra systemet.

Til sist vil jeg nevne online læring. Man kan utforske et system som lærer etter hvert som brukeren benytter systemet. For at systemet skal lære, er brukeren nødt til å kunne gi beskjed om systemet gjettet riktig eller ikke. Dette kan utforskes med talegjenkjenning og syntesert tale. Et annet alternativ er en liten touchskjerm der brukeren kan svare på om gesten ble gjettet riktig.

Multimodal interaksjon gjennom tale og gester

Det umiddelbare videre arbeid for den begrensede talegjenkjenningen virker ganske klart: å implementere *rejecting out of grammar utterances*. Slik systemet står nå vil enhver lyd gjenkjennes som et av ordene eller uttrykkene i vokabularet. Når usikkerheten om en lyd

er stor bør en null-verdi gjettes. Dermed unngår man at kremt, host, bakgrunnsstøy og pauselyder blir feilaktig gjenkjent som talekommandoer. Dette er foreløpig ikke implementert i Pocketsphinx.

Et spennende prosjekt ville vært å kombinere begrenset tale for å gi hjemmet kommandoer og kontinuerlig, naturlig tale for å spørre orakelspørsmål (som "hva er værmeldingen for neste fredag?" eller "finn en oppskrift på tacokrydder."). En idé er å la ulike gester aktivere den passende talegjenkjenningen. Ved å utføre den riktige gesten vil systemet, i stedet for den lokale talegjenkjenningen, åpne opp en kobling til en tjeneste, som Google, og lytte etter input. Enkle kommandoer for å styre hjemmet. Naturlig tale for å spørre orakel-spørsmål.

Brukertesting vil være et viktig steg videre for å bevise denne hypotesen. Vil brukere finne det naturlig å prate for å skru av og på apparatene i hjemmet? Er det fordelaktig å ha et sentralt, eller et fåtall sentrale steder å utøve styring fra?

Kombinasjoner

Når vi nå har etablert at disse sensorene kan oppnå en forståelse for gester ved hjelp av maskinlæring, dukker det opp andre bruksområder. Spesielt scenarier der det er fordelaktig å unngå fysisk kontakt er interessante. Et eksempel er toaletter og vasker, der en enkel gest kan skylle ned eller starte vannet, og andre gester kan justere vanntemperaturen. Et annet sted der hygiene er svært viktig er på sykehus. Helsepersonell kan styre ulik funksjonalitet på et sykehus, uten å komme i kontakt med knapper, skjermer og dører. Her er det store muligheter for utforskning.

Dersom man benytter systemet slik at det må både tale og gester til for å aktivere funksjonalitet, har man i praksis en 2-veis autentisering. Dette kan dermed brukes som en sikkerhetsmekanisme. En passordfrase av vilkårlig lengde sammen med en eller flere korrekte gester, kan åpne en lås eller gi tilgang til å styre funksjonaliteten i hjemmet. Dette vil være en svært trygg autentisering. Samtidig er den effektiv å utføre og tar minimalt med plass.

Et annet aktuelt applikasjonsområde å utforske er tablets. Hva kunne man fått til dersom slike sensorer ble plassert i hjørnene, på kantene eller på baksiden av en tablet? Gester

kunne brukes for å utføre vanlige funksjoner som å scrolle og å navigere mellom app-er eller nettleserfaner. Kanskje helt nye interaksjonsformer nå kunne blitt utforsket. Kunne man utnyttet abstrakte gester i luftrommet rundt tablet-en for å skape god interaksjon?

Mange manipulasjonsprogrammer i dag har store menyer med verktøy og innstillinger man skal velge i. Kunne man benyttet mer av skjermarealet til det primære utviklingsområdet ved å tilby valg av verktøy og innstillinger gjennom tale og gester?

Kontekstdrevet brukergrensesnitt

For å gi brukergrensesnittet en grad av intelligens ble det implementert et utvalg regler. Disse definerte hvilken informasjon som var av prioritet og som skulle presenteres for brukeren. En videre utvikling kunne være å la brukere legge inn egne regler, og selv definere hva som er av prioritet. Videre kunne teknikker fra maskinlæring også her benyttes for å la programmet selv lære hva brukeren foretrekker.

Å knytte brukergrensesnittet til reell sensordata ville vært et viktig steg videre. Man kunne så utforsket hvordan brukere benyttet systemet til å få svar på det de leter etter. Brukertesting av både tradisjonelle interaksjonsdrevne grensesnitt og kontekstdrevne grensesnitt, vil skape empiriske data slik at sammenlikninger og konklusjoner kan trekkes.

5.3 Konklusjon

Denne oppgaven har tatt for seg et utvalg tilnærminger til brukergrensesnitt i smarte hjem. Jeg valgte å ta utgangspunkt i en tredeling av typer programvare: programvare for informasjons, manipulasjons og kommunikasjon. Denne inndelingen gjorde at jeg kunne identifisere og kategorisere ulike aktuelle intelligente brukergrensesnitt og argumentere for hvilke brukergrensesnitt som best mulig løser brukernes problemer i smarte hjem.

Målene for oppgaven var i utgangspunktet noe høytsvevende og vanskelig å måle. Til gjengjeld var de spesifikke hypotesene til en høyere grad konkrete og målbare. I den foregående diskusjonen argumenterte jeg for hvordan hypotesene ble utfordret på en god måte. Det første målet var å benytte maskinlæring til å gi enkle sensorer utvidet funksjonalitet. Dette målet

ble fullført gjennom eksperimentene 1 og 3 der dataene fra relativt enkle gestesensorer ble tilpasset og brukt i lineære klassifiseringsalgoritmer. Resultatene var svært lovende og viser hvordan maskinlæring kan gi større funksjonalitet enn det vi kan eksplisitt programmere. Det andre målet var å eksperimentere med multimodal input. Jeg ønsket å finne en enkel, men tilstrekkelig måte å håndtere input fra gester og tale samtidig. Å benytte køer og tråder løser problemet med å håndtere asynkron input. Men det ser ikke nærmere på elementer som ble omtalt i teorien, som å forbedre forståelse eller å velge den beste inputen fra flere kanaler. På dette viset er dette målet ikke tilstrekkelig utforsket i oppgaven. Jeg endte i stedet opp med å fokusere på problemene med kontinuerlig tale og hvordan begrenset tale kan benyttes.

Fra bakgrunnsteorien lærte vi om tilnærming gjennom agenter for å håndtere grensesnittet. Min løsning er også her enklere, men i min mening ikke svakere. Å introdusere agenter kan gjøre systemet mer komplekst. Dersom hver agent er godt adskilt og all kommunikasjon foregår gjennom meldinger, minimeres kompleksiteten. Et slikt system vil være større, men vil være så enkelt som mulig. Agentene i Ishizaki (1996) virker sammenkoblede i den forstand at de selv må forholde seg til hverandre. En slik løsning er unektelig mer dynamisk og systemet kan selv komme opp med ny måter å presentere data på. Jeg er usikker på om dette vil gi tilstrekkelig gode resultater i et brukergrensesnitt for smarte hjem. Dersom målet er å vise dataene på en best mulig måte har jeg større tro på en kompetent grafisk designer, enn et sett av agenter som skal samarbeide om å vise informasjonen på en optimal måte.

Denne oppgaven har vist at intelligente brukergrensesnitt *kan* benyttes i smarte hjem og har startet en diskusjon om de *bør* benyttes. Ved å utforske maskinlærte gester, begrenset tale, håndtering av multimodalitet og dynamiske brukergrensesnitt, har jeg vist at intelligente brukergrensesnitt kan tilby en mer naturlig og effektiv interaksjon i smarte hjem.

Referanser

Augusto, J. C. and Nugent, C. D. (2006). Smart homes can be smarter. *Designing Smart Homes*, LNAI 4008:1–15.

Bonino, D. and Corno, F. (2011). What would you ask to your home if it were intelligent? exploring user expectations about next-generation homes. *JOURNAL OF AMBIENT INTELLIGENCE AND SMART ENVIRONMENTS*, 3,2:111–126.

Cook, D. J., Young, M., and Das, S. K. (2006). A multi-agent approach to controlling a smart environment. *Designing Smart Homes*, pages 165–182.

De Silva, L. C., Morikawa, C., and Petra, I. M. (2012). State of the art of smart homes. *Engineering Applications of Artificial Intelligence*, 25:1313–1321.

Dixon, C., Mahajan, R., Agarwal, S., Brush, A., Lee, B., Saroiu, S., and Bahl, P. An operating system for the home. *Microsoft Research*.

Dumas, B., Lalanne, D., and Oviatt, S. (2009). Multimodal interfaces: A survey of principles, models and frameworks. *Human Machine Interaction*, pages 3–26.

Elliott, M., Petra, M., and et al (2009). Quantifying sway through surface deflection patterns: a novel approach using distributive tactile sensing. *Procedures Inst. Mechanical Engineering, PartHJ.Eng.Med.223,7*, pages 131–136.

Farella, E., Falavigna, M., and Ricco, B. (2010). Aware and smart environments: The casattenta project. *Microelectronics Journal*, 41:697–702.

Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., and Wang, M. (2004). Proxemic interactions: The new ubicomp? *interactions*, january/february:42–50.

Intille, S. S., Larson, K., Beaudin, J. S., Tapia, E. M., Kaushik, P., Nawyn, J., and McLeish, T. (2005). The placelab: a live-in laboratory for pervasive computing research (video). *Pervasive 2005 Video Program*.

Ishizaki, S. (1996). Multiagent model of dynamic design. *CHI*, April.

Jaimes, A. and Sebe, N. (2007). Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding*, 108:116–134.

Kaufmann, M. (1998). Intelligent user interfaces: An introduction. *RUIU*.

Koskela, T. and Vaananen-Vainio-Mattila, K. (2004). Evolution towards smart home environments: empirical evaluation of three user interfaces. *Pers Ubiquit Comput*, 8:234–240.

Lieberman, H. (2009). User interface goals, ai opportunities. *AI Magazine*, Winter.

Peine, A. (2008). Technological paradigms and complex technical systems–the case of smart homes. *Research Policy*, 37-3:508–529.

Reinisch, C., Kofler, M. J., Iglesias, F., and Kastner, W. (2011). Thinkhome energy efficiency in future smart homes. *EURASIP Journal on Embedded Systems*, page 18.

Rogers, Y. (2006). Moving on from weiser's vision of calm computing: Engaging ubicomp experiences. *P. Dourish, A. Friday (Eds.) Ubicomp 2006*, pages 404–421.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, NJ, 3rd edition.

Röcker, C., Janse, M. D., Portolan, N., and Streitz, N. (2004). User requirements for intelligent home environments: A scenario-driven approach and empirical cross-cultural study. *Joint sOc-EUSAI conference*, pages 111–116.

Schneiderman, B. and Maes, P. (1997). Direct manipulation vs user interface agents. *interactions*, november + desember.

Streitz, N. A., Röcker, C., Prante, T., van Alphen, D., Stenzel, R., and Magerkurth, C. (2005). Designing smart artifacts for smart environments. *IEEE Computer*, March:41–49.

Taylor, S., Keskin, C., Hilliges, O., Izadi, S., and Helmes, J. (2014). Type-hover-swipe in 96 bytes: A motion sensing mechanical keyboard. *CHI*.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Pr, 2nd edition.

Victor, B. (2006). Magic ink: Information software and the graphical interface.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, April:94–104.

A Ressurser

Lenker

Nettsted kontekstdrevet brukergrensesnitt

https://thesmarthome.herokuapp.com/

Kildekode kontekstdrevet brukergrensesnitt

https://github.com/flaket/smarthome

Kildekode multimodal interaksjon gjennom tale og gester & kombinasjoner

https://github.com/flaket/multimodal

Kildekode gestegjenkjennelse gjennom fotodioder

https://github.com/flaket/gesture-machine-learning

B Akronymer

AI Artificial Intelligence

AmI Ambient Intelligence

API Application Programming Interface

BCI Brain-Computer Interfaces

CSS Cascading Style Sheets

DOM Document Object Model

GOMS Goals, Operators, Methods and Selection rules

HCI Human Computer Interfaces

HMM Hidden Markov Model

HTML HyperText Markup Language

IKT Informasjons- og kommunikasjonsteknologi

IoT Internet of Things

IUI Intelligent User Interface

I2C Inter-Intergrated Circuit

JVM Java Virtual Machine

kNN k Nearest Neighbors

LED Light Emitting Diode

FIFO First-In-First-Out