BILDER OG STEDSINFORMASJON

INF-3981 Masteroppgave i informatikk

Kai Arne Bjørnenak
Institutt for informatikk
Fakultet for matematikk og realfag
Universitetet i Tromsø, Norge

Sammendrag

Enorme mengder informasjon er tilgjengelig på internett for browsing og søk for folk på jakt etter informasjon. En økende del av denne informasjonsmengden er bilder, video og audio. Disse media-typene er ikke så lette å kategorisere og beskrive som en tekstlig ressurs. Vi trenger derfor nye metoder for å takle utfordringene denne fremveksten av nye media gir oss.

Digitalt fotografi har hatt en voldsom vekst de siste 10 årene og terskelen for å legge ut bilder allment tilgjengelig er lavere enn før. For å kunne organisere disse bildesamlingene for å gjøre dem tilgjengelig for søk må man kunne beskrive innholdet i bildene. Dette er ikke en lett oppgave, manuell annotering til bilder er langtekkelig og uøkonomisk og teknologien for å beskrive det visuelle innholdet i bildene er enda umoden.

Samtidig ser vi at en god del informasjon kan utledes om bildet gjennom kontekst, informasjon som er situasjonsbetinget omstendighetene når bildet ble tatt. Denne avhandlingen undersøker et aspekt ved kontekst til bilder, nemlig konteksten som har med hvor bildet ble tatt, eller stedet. Vi undersøker spesielt hvordan man kan koble en posisjonsangivelse gitt med koordinater med et lokalt stedsnavn som er lettere å forstå for mennesker.

En prototype er implementert som baserer seg på et selvlaget datasett og bildesamling. Resultatene fra eksperimentene med prototypen og fra arbeidet med denne avhandlingen antyder at lokasjon kan være et godt hjelpemiddel til å generere metadata til bilder.

Takk til

Jeg vil takke min veileder Randi Karlsen for inspirasjon og god oppfølging gjennom hele arbeidet med denne avhandlingen, jeg vil også gjerne takke mine foreldre for deres støtte gjennom studietiden min.

Innhold

Figurer

Kapittel 1

Introduksjon

Fokuset til denne masteroppgaven er å se på metoder for å finne og anvende stedsinformasjon som metadata til bilder. Denne oppgaven er en del av CAIM-prosjektet ved Universitetet i Tromsø.

1.1 Motivasjon

Det finnes en enorm mengde med informasjon på nettet, men vi har problemer med å aksessere denne informasjonen på grunn av vår manglende evne til å gjøre denne informasjonen tilgjengelig. Daglig legges det ut enorme mengder med ny informasjon i form av tekst, bilder, video og audio ut på nettet, for at vi skal kunne ha mulighet til å effektivt kunne utnytte denne informasjonen trenger vi nye metoder for å aksessere den.

Fremveksten av digitalkameraer gjør at bilder, både stillbilder og levende bilder utgjør en større og større del av informasjonen som er tilgjengelig på nett, men denne visuelle informasjonen er som regel bare tilgjengelig gjennom tekstbaserte søk som er avhengig av personen som søker sin evne til å spesifisere søket. For at denne metoden skal være effektiv kreves det at bildene er analysert for innhold og annotert deretter. Annotering av store bildesamlinger er en tidkrevende prosess og ofte er ikke teksten assosiert med bildet annet enn med en tittel eller omkringliggende tekst som bruker bildet som en illustrasjon. For å gjøre søk etter bilder mere effektivt trenger man annoteringsteknikker som gjør informasjon som ligger i bildet tilgjengelig for søk, samtidig vil dette innebære at nødvendigheten for samle inn og organisere metadata vil bli viktigere.

CAIM (Context Aware Image Management) mener at at et bilde kan være assosiert med flere ulike kontekster og at kjennskap til disse kontekstene kan forbedre kvaliteten til søk etter bilder. Kontekstsensitivitet kan bli brukt for å identifisere bildesemantikk og sammenhenger og kan bli brukt for å minske det semantiske skillet innenfor *information retrieval*. Målsetningen

til CAIM er å redusere brukerens arbeid for å anskaffe seg relevant informasjon i forhold til innsatsen som må til med dagens søkemotorer.

Denne avhandlingen vil fokusere på kontekst i form av lokasjon. De fleste mennesker har en klar formening om hva som ligger i begrepet sted eller lokasjon, for de fleste er dette et avgrenset geografisk område som har et navn. Det viser seg at lokasjon kan være et godt hjelpemiddel for *image retrieval*, og for å organisere bildesamlinger. Kontekst som baserer seg på sted representerer informasjon som sier noe om opprinnelsen til bildet, dette kan være relasjoner mellom bildet og objekter i bildet, stedet der bildet ble tatt og hendelser som er representert i bildet.

1.2 Avhandlingens bidrag

For å anvende lokasjon til å innhente informasjon om bilder trenger man først et begrep om hva et sted er. For å samle inn geografisk lokasjonsinformasjon anvender vi en GPS som kan gi en relativt eksakt geografisk posisjon, men denne måten å angi en posisjon på er ikke spesielt brukervennlig, derfor vil denne avhandlingen se på en metode for å oversette geografiske posisjoner i form av GPS-koordinater om til stedsnavn som er forståelige for mennesker. For å teste om dette er gjennomførbart har vi laget en enkel prototype som skal utføre nettopp denne oversettelsen fra koordinater til stedsnavn.

Prototypen fungerer som et slags elektronisk oppslagsverk hvor man kan sende inn en geografisk posisjon, prototypen vil deretter gjøre et oppslag og gitt at posisjonen finnes innenfor det området prototypen har data for og den er innenfor en akseptabel distanse fra et kjent sted innenfor dette området så vil man få ut et stedsnavn. På grunn av høyt tidspress er arbeidet på prototypen til en viss grad neglisjert til fordel for denne teksten.

Denne avhandlingen har et litt annet fokus enn andre sammenlignbare prosjekt av denne typen da den konsentrerer seg om kontekst-informasjon fra sted med vekt på lokale referanser.

1.3 Mål

Målet for denne oppgaven er å kunne knytte kontekstinformasjon til en bildesamling i form av informasjon om stedet der bildene ble tatt. På denne måten kan man gjøre automatiske annoteringer til innholdet i et bilde eller en bildesamling. Håndholdte GPS-mottakere gjør det mulig å knytte en relativt nøyaktig posisjon til et bilde tatt på et distinkt tidspunkt. Denne oppgaven vil se på måter å oversette en posisjon til stedsnavn/navn

på objekter i bildet som har mening for mennesker og måter å søke etter informasjon relatert til stedet/objektet som kan si noe om innholdet i bildet.

1.4 Metode

Metoden brukt i denne avhandlingen er en prototyping-modell hvor man først har funnet ut av hvilke overordnede mål og konsepter som er viktige for denne typen systemer, basert på disse er det så blitt bygd en prototype som skal vise at et subsett av konspetene er gyldige og at de lar seg implementere. Således fungerer prototypen i denne avhandlingen som en *proof of concept* på at en sentral del av det foreslåtte systemet faktisk virker.

1.5 Oppbygning av denne avhandlingen

Kapittel 2 ser på tidligere arbeid innenfor feltet, teori og bakgrunn og relaterte prosjekter. Teori og bakgrunn for innhenting av informasjon og bilder, kontekst relatert til bilder og hva man kan hente ut fra georefererte bilder finnes i dette kapittelet.

Kapittel 3 presenterer de forskjellige teknologiene for å gjøre lokalisering og også en del om metadata og formater for metadata. GPS, GSM og trådløse lokalisering sine fordeler og ulemper blir redegjort for.

Kapittel 4 foklarer de grunnleggende forutsetninger, problemstillinger og konsepter som må være på plass før et system som kan gjøre lokalisering for georefererte bilder kan realiseres. Noen av problemstillingene er innsamling av data, hvordan lokasjoner blir representert i systemet, hvordam man tilordner stedsnavn til posisjoner og organisering og lagring av metadata.

Kapittel 5 beskriver design, implementasjon og testing av prototypen som bygger på noen av konseptene beskrevet i kapittel 4.

Kapittel 6 diskuterer resultatene fra prototypen, fremtidig arbeid, forbedringer av systemet og hva man har lært av arbeidet med prototypen og denne avhandlingen.

Kapittel 7 Konkluderer og avslutter denne avhandlingen.

Kapittel 2

Teori og relaterte prosjekter

Dette kapittelet vil se på teori og forskning relatert til bilder og stedsinformasjon og andre prosjekter som arbeider med samme tema. Vi vil definere hva som ligger i begrepet kontekst og kontekstsensitive applikasjoner og hva som menes med det semantiske skillet.

2.1 Innhenting av informasjon

Mye forskning har gått med til å beskrive måter å innhente informasjon og bilder fra gitte datasett på. Spesielt med tanke på internett finnes det mye forskning som beskriver metoder for innhenting av informasjon. I internettets spede begynnelse var disse metodene spesielt rettet mot å finne relevante tekstlige ressurser, gjerne i form av nettsider og dokumenter. Etter hvert som verdensveven utviklet seg og båndbredde ut til brukeren vokste så man mer og mer til andre media-former på nettet. Forskjellige media ble en stadig viktigere del av innholdet som brukere ville ha tak i og like viktig ble det å få bedre metoder for å lokalisere disse i det globale nettet.

"The hallmark of a good retrieval system is its ability to respond to a user's queries and present results in a desired fashion" [19]. Med dette sitatet får vi en pekepinn om hva som er viktig å tenke på når man gjør søk etter informasjon. Første prioritet er selvfølgelig å få relevante resultater fra søket, men hvis denne informasjonen ikke blir presentert på en hensiktsfull måte så mister den mye av sin verdi for brukeren.De best kjente eksemplene på systemer for informasjonsinnhenting er de store internett-søkemotorene, for eksempel Google og Yahoo [12][41]. Disse har relativt nylig kommet opp med egne søk for audio, video og bilder. Utfordringen med å komme opp med relevante søkeresultater for disse mediatypene ligger i å mestre det semantiske skillet. I dag brukes som regel bare tekstlige ressurser slik som bildetittel og omkringliggende tekst for å finne relevante resultater til søket uavhengig av om man søker etter bilder, video eller audio.

2.2 Innhenting av bilder

En av utfordringene innen feltet informasjonsinnhenting er bildesøk. Et bilde er mye vanskeligere å kategorisere enn en tekst, noe man finner igjen i uttrykket "et bilde sier mer enn tusen ord". Når man skal søke etter bilder vil man ha bilder som samsvarer med søkekriteriene, men dette er vanskelig å få til, for hvordan henter man ut informasjon om innholdet i et bilde? Tilhørende tekst og bildetittel er gode hjelpemidler, men gjerne ikke tilstrekkelig. Det man trenger er teknikker for å beskrive innhold i et bilde, en objektiv standard, om mulig, for å spesifisere søk i bildesamlinger og for lagring/organisering av kontekstuelle metadata til bildene.

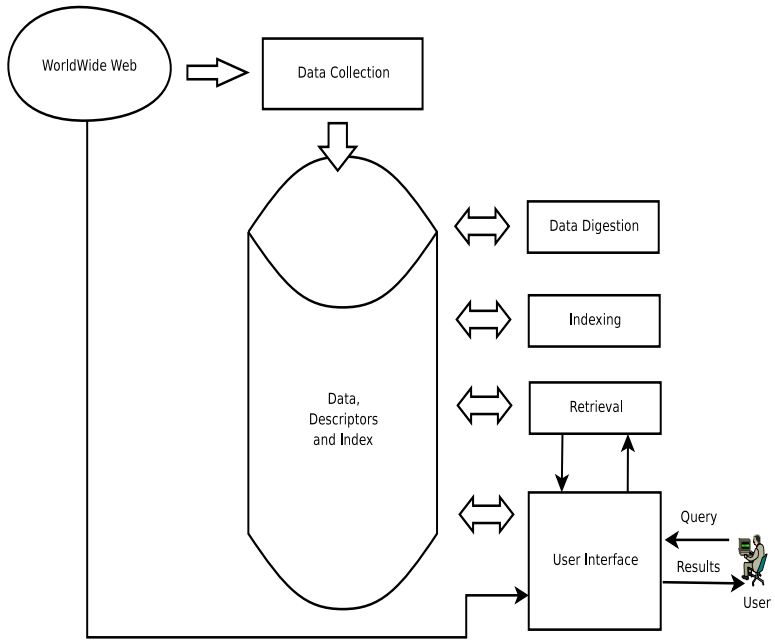
En av utfordringene ligger i å la brukeren kunne spesifisere hvilket innhold han/hun er ute etter i bildet. Hvordan vil en bruker spesifisere et søk? For å ta et enkelt eksempel, la oss si at en bruker søker etter bilder av hunder, men en hund kan være så mangt, valper, tisper, ulv, schaefer, puddel osv, alle tilfredsstiller det overordnede søkekriteriet. Det ideelle hadde vært om en datamaskin kunne "se" på et bilde og med en gang avgjøre om det er en hund i bildet eller til og med avgjøre om det var hunden som var fokuset for bildet. Et menneske kan med letthet avgjøre om det er en hund på et bilde og i de fleste tilfeller avgjøre hvor fokuset i bildet ligger. Det er dette som er fokus for forsking innenfor innholdsbaserte systemer innenfor bildesøk. For å ta enda et eksempel med hunder som illustrerer situasjonen i dag, et enkelt søk med google images [1] etter "dogs" gir rett nok masse bilder av hunder, men det gir også bilder av en sau, et hundehjerte og Claudia Schiffer. Spennvidden i resultatene gir en pekepinn om svakheten ved dagens teknikker for innhenting av bilder. Forskjellige metoder for innhenting av bilder eksisterer, mange av disse konsentrerer seg om innhold i bildene [32]. Det har vært flere forsøk på å lage dedikerte søkemotorer for bilder og andre media på nettet, blant annet `ImageRover` [35], `WebSeek` [36] og `Atlas WISE` [23]. Figur 2.1 gir en oversikt over hva en søkemotor trenger å gjøre.

2.2.1 Tekstbasert innhenting av bilder

Tekstbasert innhenting av bilder er den teknologien vi har i dag for å gjennomføre søk etter bilder på internett. Disse søkemotorene virker ved at de periodisk gjennomsøker nettet og lagrer tekst som de finner. Denne informasjonen må deretter analyseres og klassifiseres, slik at informasjonen kan bli gjort tilgjengelig for søk. Periodiske søk er nødvendig siden nettet stadig endrer seg, med dokumenter og media som blir lagt til, endret eller slettet hvert eneste sekund.

Hvilke tekstlige ressurser er det så man kan utnytte når man ønsker å søke etter et bilde på nett. Bildetittel og omkringliggende tekst har allerede blitt nevnt, men det er andre tekstlige attributter knyttet til bildet som kan

[1] images.google.com



Figur 2.1: Generell struktur til en søkemotor for bilder [24]

utnyttes. Noen eksempler: bilde-etikketter, tittel på websiden, filnavnet til bildet, hyperlinker til bildet og titler som ligger i kilden til nettsiden. Disse attributtene er ikke alltid relatert til bildet, noen søkemotorer legger derfor forskjellig vekt på teksten avhengig av hvor den ble funnet på nettsiden, om teksten forekommer hyppig og posisjonen til teksten i forhold til bildet for å forbedre sjansen for relevante treff.

Ved å bruke tekst som er relatert til bildet støter man på problemet med subjektivitet. Hvor relevant er teksten som følger med bildet på en nettside. Har man samme oppfatning av hva som er fremstående attributter/seksjoner i bildet som personen eller personene som har laget nettsiden. Her kan verktøy for automatisk bildeannotering og standarder for metadata hjelpe. Man kan jo også spørre seg om det å basere seg helt på tekstlige ressurser er nok. Et bilde består vanligvis av mange komponenter, og noen som skriver en tekst om bildet vil kanskje konsentrere seg om noen få komponenter i bildet. Det er også noen egenskaper ved bilder som ikke egner seg så godt til å beskrives med tekst. Dette kan være ukjente objekter, komposisjon, teksturer og andre [24]. Rent visuelle egenskaper ved bildet kan komplementere informasjonen man får ut fra attributter basert på tekst. Her kommer *Content-Based Image Retrieval* inn i bildet.

2.2.2 Innholdsbasert innhenting av bilder

Også kjent under den engelske forkortelsen CBIR, CBIR bruker de visuelle karakteristikkene til bildet for å kategorisere det og generere metadata som beskriver disse egenskapene. Med dagens teknologi er det mulig å hente ut en rekke visuelle kjennetegn fra et bilde, man kan blant annet kjenne igjen farger, orientering, kanter, teksturer, mønstre og former. Et bilde kan inneholde svært mye informasjon slik som former, farger, kanter og så videre, av den grunn er det vanskelig å kategorisere det etter disse kriteriene. Det kan være at man er bedre tjent med spesialiserte CBIR-systemer i stedet for et system som prøver å hente ut all mulig visuell informasjon. Systemer som ser etter et relevant subsett av de mulige visuelle karakteristikkene til et bilde. CBIR er et veldig aktivt forskningsfelt innenfor *information retrieval* noe som også gjenspeiles i mengden litteratur om emnet som er kommet de siste årene [32][42].

2.2.3 Automatisk annotering til bilder

Med automatisk bildeannotering mener vi metoder for å la et program lage og lagre metadata om et bilde. Det er en viktig del av et system for innhenting av bilder. Annoterte bilder gir en bruker muligheten til å spesifisere søk på en mere naturlig måte, i motsetning til mange av de innholdsbaserte *image retrieval* systemene i dag der brukeren ofte må bruke visuelle konseptet slik som former og farger, eller ved å bruke eksempelbilder for å spesfisere et søk. Annotering gjør det lettere å representere semantiske konsepter [20][2][26].

Manuell annotering til bilder har vært i bruk lenge, men denne måten å gjøre det på har to distinkte ulemper, den ene er at manuell annotering er tidkrevende og kostbart, den andre er at menneskers oppfatinger er subjektive og det en person ser som de viktigste særpregene i et bilde trenger ikke være det samme som en annen person velger å vektlegge. En tredje kritikk som har vært reist mot tekstlige beskrivelser av bilder er at det kan være særtrekk i bildet som ikke egner seg til å bli beskrevet med tekst [21]. Det er særlig de to første punktene som forskning innenfor automatisk bildeannotering forsøker å løse.

2.3 Det semantiske skillet

Det semantiske skillet er ikke bare et problem innenfor innhenting av informasjon, men innenfor alle felt hvor konsepter på et høyt nivå må konsolideres med konsepter på et lavt nivå, for eksempel kan man ha beskrivelse av hva et program skal gjøre skrevet i et naturlig språk som kan komme i konflikt med det man faktisk kan gjøre med et programmeringsspråk.

For å finne bilder som er relevante til et søk trenger man teknikker som kan si noe om innholdet i bildet. Man ser at metodene som man har i dag er for unøyaktige for å kunne gi en presis representasjon av bildet. Vi sier at det finnes et semantisk skille [42] mellom informasjonen vi kan hente ut fra et bilde ved automatiske teknikker og de høynivå konseptene som en bruker vil anvende for å beskrive bildet i et søk. Det finnes metoder for å komme seg rundt denne problemstillingen på, en er å anvende manuell annotering, men på grunn av størrelsen på bilde og multimediasamlinger er dette som regel ikke praktisk gjennomførbart på grunn av hensyn til tid og økonomi. En annen metode man kan bruke for å finne informasjon om bildet er å benytte seg av konteksten til bildet. Med denne kontekstuelle informasjonen er det mulig å lage relevante automatiske annoteringer til bildet.

En annen utfordring med det semantiske skillet er at man ikke alltid vet hva et ord er tenkt å bety i en gitt sammenheng, dette er referert til som *Synonymy* og *Polysemy*, det første refererer til at man ofte har flere ord for å representere den samme tingen, eksempel fjell, berg, klippe, haug som i gitte situasjoner alle kan brukes om samme objekt, det andre refererer til at man har et ord som har flere betydninger, se på klippe i det forrige eksempelet, mener man en klippe som stikker ut i sjøen, eller mener man klippe som det man gjør med en saks.

2.4 Kontekst og kontekstsensitivitet

Vi vil definere kontekst som enhver informasjon som kan bli brukt til å karakterisere situasjonen til en entitet, der en entitet kan være en person, en plass, et fysisk objekt eller et virtuelt objekt [1]. Så kontekst kan være et veldig omfattende begrep og er betinget av situasjonen til objektet. Dette gjør det vanskelig å lage kontekstsensitive applikasjoner som klarer å gjøre nytte av kontekstinformasjon i alle situasjoner.

For kontekstsensitive applikasjoner ligger utfordringen i å lage et system som er kontinuerlig bevisst over objektets skiftende kontekst [33] og som klarer å tilpasse seg denne. Det vanligste eksempelet her er i form av en elektronisk turistguide, der brukeren har på seg flere sensorer som registrerer kontekst og som kontinuerlig henter ned informasjon som er relevant til konteksten til brukeren[4]. Langt de fleste kontekstsensitive applikasjoner dreier seg om å bruke kontekst til å innhente relevant informasjon og det er også denne anvendelsen vi vil se på her. Forskjellige typer kontekstsensitive applikasjoner eksisterer, der fire hovedtyper er:

- Seleksjon ved nærhet - Denne typen applikasjoner ser på hvilke ressurser som er i nærheten av brukeren og gir lettere tilgang til disse. For eksempel, er en bruker på et møte og brukeren trenger å gi et dokument lagret på et håndholdt apparat til alle på møtet, en eller

flere sensorer vil oppfatte situasjonen og vil under situasjonen "møte" gi brukeren tilgang til alle de som er tilstede i møterommet.

- Automatisk kontekstuell rekonfigurasjon - Dette vil si at en kontekstsensitiv applikasjon kan bytte ut komponenter, legge til nye eller endre forbindelsen mellom komponenter. Stikkordet her er automatikk, at en applikasjon gjenkjenner at konteksten er endret såpass at nye komponenter trengs for å ta seg den nye situasjonen. Igjen er eksempelet med møterommet gyldig, sensorer vil gjenkjenne at man har beveget seg fra en kontor-kontekst til et møte og vil reagere deretter.

- Kontekstuell informasjon og kommandoer - Brukeren får stadig oppdatert informasjon basert på sin kontekst, for eksempel når brukeren forflytter seg fra et rom til et annet, la oss si fra et kontor til en forelesningssal, en kommando for å hente ut forelesningsnotatene kan da komme opp eller informasjon om hvor forelesninga skal være.

- Kontekst-utløste handlinger - Dette er enkle handlinger som er utløst når en kontekstuell betingelse blir oppfylt, for å fortsette med det foregående eksempelet, når foreleseren går inn i forelesningssalen kan en prosjektør slås på og vise dagens forelesning på lerretet.

En kontekstsensitiv applikasjon kan selvfølgelig inkorporere flere eller alle av disse typene, og i mange situasjoner er nok også dette ønskelig.

2.5 Bilder og sted

Hva er et sted? Vi vil definere et sted som en posisjon eller et punkt i det virkelige rom som er uttrykt relativt til et annet punkt eller posisjon. Vi kan snakke om to forskjellige måter å uttrykke dette på, nemlig absolutt posisjon og relativ posisjon.

2.5.1 Absolutt posisjon

Med absolutt posisjon har man distinkt definerte punkt på planeten der punktet virkelig er, relativt til en georomlig (*geospatial*) modell. For eksempel er breddegrad, lengdegrad parene som man er vant med å bruke ofte uttrykt i forhold til *World Geodetic System* også kjent som `WGS84` [40]. Et eksempel på en absolutt posisjon kan være 6940.887 N og 1858.334 Ø som angir et eksakt punkt og som befinner seg ved Teorifagbygget ved Universitetet i Tromsø. Dette er en geometrisk fremstilling av et sted der man bruker et sett av tall definert av en georomlig modell for å gi en referanse til stedet.

2.5.2 Relativ posisjon

En relativ posisjon vil si at man definerer hvor en posisjon/sted er i forhold til en annen virkelige posisjon. Man kan for eksempel si at punkt a ligger 100 meter rett nord for punkt b, og da har man en referanse som kan anvendes til å si noe om punktet. Dette kan sees på som en symbolske fremstilling av et sted, der man bruker en tekstlig beskrivelse for å gi en referanse til stedet [3]. Dette kalles landemerke-basert posisjonering og er knyttet til posisjonering med for eksempel mobilmaster, hvor man avgjør posisjonen i forhold til et fast punkt i terrenget [22].

2.5.3 Sted

Hvordan skal man klare å knytte kontekst-informasjon i form av lokasjon til et bilde? GPS-moduler til kameraer har vært tilgjengelig for en stund nå, noen kommer med en laser avstandsmåler og kompass så man skal kunne finne avstand og retning til objektet man tar bildet av. Men det er først nylig at GPS-moduler til kameraer har blitt allment tilgjengelig, for eksempel lanserte en av de største produsentene av forbrukerelektronikk nylig en GPS-modul til sin mest populære digitalkamera-serie. Også populære webtjenester som `flickr` [11] og `mappr!` [27] lar allerede brukere knytte bilder til steder ved å manuelt droppe bilder på et kart og slik knytte bildet til et sted. Bilder som inneholder informasjon om hvor de ble tatt kalles gjerne georefererte bilder.

Typer av lokasjonssystemer

Grafiske grensesnitt for å organisere bilder etter stedet de ble tatt kan deles opp i to hovedtyper:

- Kartbaserte - Bruken av kart for å gi en visuell presentasjon av geografisk data har vært i bruk lenge og mange lokasjonssystemer anvender kart for sine tjenester, Google Maps [14] er kanskje den mest kjente. En av fordelene med kartbaserte tjenester er at de er lette å navigere og intuitive i bruk, klikk på et punkt for å sentrere på et punkt, en slider eller dobbeltklikk zoomer inn. Noen tjenester tilbyr dra og dropp for å gjøre bilder georefererte, andre kan sette inn georefererte bilder inn på riktig plass på et kart basert på posisjonen som ligger registrert til bildet, se Toyama et al [37] for en diskusjon av måter å gjøre dette på. Men kartbaserte systemer er kanskje ikke den beste veien å gå for å organisere bilder, skjermstørrelsen er begrenset og bilder/thumbnails tar stor plass på skjermen, med mange bilder tatt på samme sted blir det fort rotete og uoversiktlig, for håndholdte apparater blir problemet bare større.

- Tekstbaserte - Systemer som baserer seg på interaksjon med dataen ved hjelp av tekst har ikke det samme problemet som med kartbaserte systemer. Disse fungerer godt på små skjermer og adopterer som regel en hierarkisk trestruktur for å representere forskjellige regionale nivåer

2.5.4 Tid

Å inkludere tid i metadataene til et bilde er allerede vanlig og er nyttig for å sortere bildene i en bildesamling kronologisk og nå når tilgang til posisjonen der bildet ble tatt begynner å bli vanligere åpner det for en ny type sortering sammen med sted. Her kommer ideen om hendelser inn, det er to måter man kan se på hendelser, den ene som en virkelig hendelse som man har tatt bilde av, for eksempel et bryllup, nasjonaldagen osv. Den andre måten er bestemt av metadata. Her bruker man en algoritme som ser på metadataene til en bildesamling og sorterer bildene basert på tid og sted. Mange bilder tatt over et visst tidsrom på samme sted blir definert som en hendelse [22].

2.5.5 Informasjon fra georefererte bilder

En av de store fordelene med å ha tid og sted integrert som metadata i et bilde er at de kan fungere som generatorer for enda mer kontekstinformasjon [28]. Med tid og sted på plass kan man for eksempel finne ut om det var dagslys når bildet ble tatt, hvordan været var på det stedet, flo eller fjære, høyde over havet og så videre. Alt dette kan finnes ut av ved søk på nettet, arkivdata fra værstasjoner er ofte tilgjengelig på nett [8], tabeller over dagslys er også tilgjengelige [5]. Andre typer metadata som kan genereres ut i fra tid og sted er elevasjon, informasjon om stedet man tok bilde fra og eventuelt stedet man tok bilde av, hendelser på det stedet man tok bilde av, om man skulle befinne seg i Rio de Janeiro under karnevalet kunne man tenke seg at bildene derifra ble samlet under "Karneval i Rio - 2006", noe som kan gi god dytt til hukommelsen i stedet for bare en dato-tids-angivelse eller enda verre en tittel som IMG_021.JPG som ikke forteller noen ting om bildet. Browsing av bilder basert på assosierte metadata er et nøkkelkonsept her.

2.6 Relaterte prosjekter

Denne delen vil gi en kort gjennomgang av andre prosjekter som har benyttet seg av kontekstinformasjonen som kan hentes ut i fra koblingen til et sted eller en posisjon.

2.6.1 WWMX

I deres artikkel om the `World Wide Media eXchange` [37] beskriver Toyama med flere et system som bruker geografiske steds-annotasjoner til digitale bilder. Deres system bruker lokasjon som et utgangspunkt for å samle inn kontekstdata til bildene, organisere bildene og presentere dem for brukeren. De bruker disse teknikkene for å koble lokasjonsinfo til bildene:

- Manuell annotering - Skriver inn lokasjonsinfo for hånd
- Bildeheader - Vil si at kameraet er koblet til et apparat som legger lokasjonsinfo direkte inn i bildeheaderen. Som regel en GPS-koblet til kameraet og som legger koordinater inn i EXIF-headeren
- Lokasjons-sensitivt apparat - Et apparat, f.eks en GPS-mottaker, som man kan samkjøre med bildene etterpå.

WWMX indekserer bilder på mange forskjellige måter, blant annet etter tid, eier, dimensjoner og så videre, men de argumenterer for at lokasjon er den viktigste av dem. Hvis man vet hvor et bilde ble tatt er det med en gang mye kontekstuell informasjon man kan hente inn om bildet.

Artikkelen diskuterer også måter å representere lokasjon på og problemet med presisjon. *WWMX* bruker vanlige lengegrad/breddegrad for å representere lokasjoner. Denne måten er veldig utbredt innenfor `GIS`-miljøet og det burde derfor være en del tilgjengelige datasett. Videre beskriver de en måte å oppnå presisjon på når man skal plassere bilder på de geografiske lokasjonene. For å kunne presentere geografiske lokasjoner basert på koordinater i forhold til et kart trenger man en data-struktur som kan gi en ide om den fysiske regionen. *WWMX* bruker en 2Dstruktur kalt `area` som beskriver en unik region på en 3D globus. Denne regionen kan deles inn i mindre regioner avhengig av hvilken presisjon man ønsker.

2.6.2 Placelab

Schilit et al [34] foreslår å bruke trådløse aksesspunkter til posisjonering og argumenterer med at i mange tettbebygde strøk gjør fremveksten av trådløse nett det mulig å beregne posisjoner med omtrent samme nøyaktighet som GPS. De har også laget en applikasjon *Place Lab* for å demonstrere prinsippet. En annen fordel med å bruke aksesspunkter til posisjonering er at de fungerer innendørs også. Den mest åpenbare ulempen ved metoden er selvfølgelig at trådløse aksesspunkter ikke er tilgjengelige overalt slik som GPS. Det kan derfor være en ide å kombinere teknologiene for å sikre dekning innendørs, utendørs og i omgivelser der trådløst nett ikke er utbygget.

2.6.3 PhotoCompass

PhotoCompass bruker sted og tid for å automatisk organisere digitale fotosamlinger i herarkier [29]. Disse hierarkiene er ment å lette arbeidet for en bruker når han eller hun ønsker å finne bestemte bilder fra fotosamlingen. De argumenterer for at stedsnavn er et sterkt hjelpemiddel for å huske tidligere hendelser og at den også er et nyttig hjelpemiddel sammen med tid for å organisere og presentere bildesamlinger. For å nyttigjøre seg av denne sammenhengen må bildene være merket med tids og stedsangivelser.

For å organisere en bildesamling trenger *PhotoCompass* å gjøre to ting, den må gruppere bildene inn i distinkte hendelser og geografiske lokasjoner, deretter foreslår den passende stedsnavn for gruppene. Navngivingen kan for eksempel gjøres med *NameSet* [30].

2.6.4 NameSet

NameSet er designet for å være et system som oversetter et sett av geografiske koordinater til stedsnavn [30]. Det er i hovedsak to bruksområder for applikasjonen, den ene er å gi en bruker en presentasjon av et sett med geografiske koordinater, den andre er å generere tekst utifra koordinatene for å kunne innhente informasjon. Motivasjonen for arbeidet med *NameSet* er den samme som denne oppgaven, at man noen ganger trenger å beskrive et sett med geografiske koordinater med stedsnavn som er forståelige for mennesker. Denne måten å representere et sett med geografiske koordinater på tar mindre plass enn et kartutsnitt, dette gjør den hendig å bruke på mobile enheter. *NameSet* ble brukt i samband med *PhotoCompass* for å komme opp med stedsnavn som ble brukt i steds-hierarkiet som *PhotoCompass* bruker for å organisere en bildesamling.

For å finne stedsnavn anvender *NameSet* seg av et kommersielt datasett som knytter sammen koordinater med navn på stedsnavn. Dette datasettet kan brukes til å avgjøre hvilket land koordinatene er i, og for USA kan de si i hvilken stat, by og/eller park som inneholder koordinatene.

NameSet består av tre distinkte prosesserings-steg, som skal sikre at den kommer opp med stedsnavn som ikke bare er riktige men som også er kjente for brukeren. I det første steget finner systemet stedsnavn for koordinatene som er gitt, for eksempel byer, parker, skoger eller stater. Parallelt med dette prøver systemet også å finne nærliggende steder som kan antas å være "kjente", sånn som store byer, kjente landemerker eller parker og lignende. Disse landemerkene trenger ikke være angitt med koordinater i settet, men er der for at brukeren skal ha en referanse til noe som kan være et lite kjent stedsnavn. Det siste steget i prosessen er å bestemme hvilket av stedsnavnene og de nærliggende landemerkene som skal brukes for å navngi stedet. For å ta et lokalt eksempel så kunne systemet ha funnet at en av koordinatene i settet korresponderte med stedsnavnet Nordkjosbotn, den ville da velge

Nordkjosbotn som stedsnavn, men siden Nordkjosbotn antakelig ikke ville være kjent som sted utenfor Nord-Norge ville den legge til "8 mil sør for Tromsø" i navnet.

2.6.5 Traces of locations

I en artikkel [22] fra Universitetet i Washington diskuterer Kang et al problemet med å gjøre om posisjoner til stedsnavn som mennesker kan forholde seg til. Ideen er at steder representert ved koordinater eller landemerker ikke er særlig brukbare for en enkelt bruker. De anvender Place Lab for å hente ut sporlogger av hvor man har vært. Problemet de så ser på i deres artikkel er hvordan man kan hente ut viktige lokasjoner for en bruker fra en slik logg. Viktige lokasjoner kan for eksempel være "jobben", "hjemme" og lignende. Forskjellige algoritmer for dette blir foreslått og diskutert. Blant annet foreslår de bruk av tidsbaserte kluster-algoritmer. Dette er algoritmer som ser på hvor lenge man befant seg på et visst sted, og denne informasjonen kan man så bruke til å avgjøre om stedet er signifikant for brukeren.

2.7 Personvern

Det finnes noen bekymringer knyttet til personvern når man snakker om tjenester som tilbyr lokaliserings-info for enkeltpersoner. Er det noen mulighet for at uvedkommende kan få tilgang til data om hvor brukeren har vært? Dette er reelle bekymringer, men samtidig skal man være klar over at det ofte er et personlig valg bak det å gjøre bilder offentlig tilgjengelig på nettet. Mange ønsker å dele opplevelser med venner og familie ved legge ut bilder på personlige nettsider, det er da litt urimelig å forvente at disse bildene ikke vil sprees. Men ved stigende bruk av metadata tilknyttet bilder må man også være bevisst at mengden informasjon som uønskede aktører kan få tak ved hjelp av disse øker.

Kapittel 3

Teknologier

Dette kapittelet vil gi et overblikk over de forskjellige lokaliserings-teknologiene som er i bruk i dag og fordeler og ulemper med å bruke dem. Vi vil også se på noen standarder for lagring av metadata.

3.1 Metadata

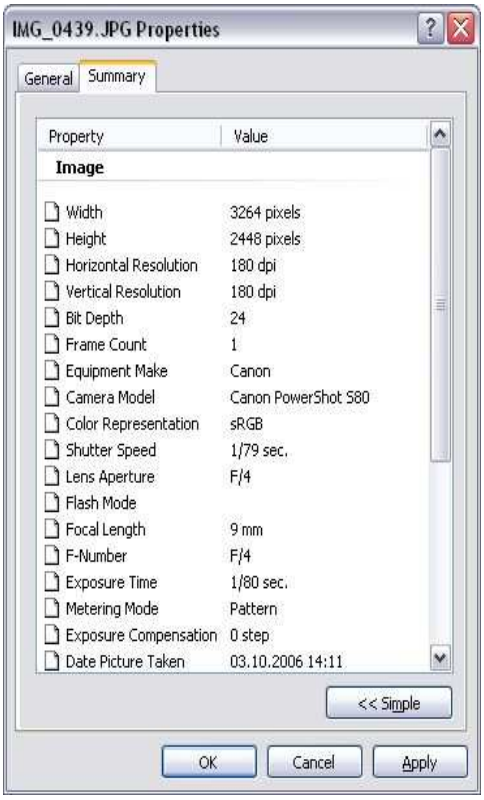
For å kunne få mest mulig nytte ut av metadata som er knyttet til et bilde er det ønskelig at man har teknikker for å spesifisere hva disse betyr i relasjon til ressursen de beskriver. `Exif` er en standard for metadata som ellerede er i bruk og to av de mest kjente initiativene innenfor metadata-spesifikasjoner er per i dag `Dublin Core` og `RDF`.

3.1.1 Exif

Exif - Exchangeable Image File Format er et filformat som mesteparten av verdens digitalkameraprodusenter støtter [10][9]. Det er et format for bildedata og bildemetadata. Det er derfor mulig å lagre mange forskjellige opplysninger om bildet. Noen av de viktigste er:

- Dato og tid
- Kamerainnstillinger
- Lokasjonsinformasjon
- Beskrivelser og opphavsrett

Metadataene trenger ikke nødvendigvis bli lagt til av kameraet når bildet blir tatt, men blir ofte lagt til i ettertid, som for eksempel informasjon om opphavsrett eller GPS-koordinater hvis kameraet ikke er GPS-aktivert. Figur 3.1 viser et subsett av Exif-data fra `egenskaper`-dialogen til et bilde i Windows XP.



Figur 3.1: I Windows XP kan man se et subsett av Exif-metadata som er lagret i et bilde

3.1.2 Dublin Core Metadata Initiative

Dublin Core er et forsøk på å lage en metadata-standard og å utvikle spesialiserte ordlister for å uttrykke metadata [6]. En stor del av arbeidet deres går ut på å utarbeide et sett av betegnelser for metadata [7]. Disse betegnelsene beskriver en type metadata som er knyttet til et objekt. Hver betegnelse har sin unike definisjon. Se figur 3.2 for eksempel på hvordan en term er definert.

Noen termer kan også ha andre typer attributter knyttet til seg for mer informasjon. En av fordelene med å definere de forskjellige typene metadata på denne måten er at de har en presis og entydig betydnig.

3.1.3 RDF

Resource Description Framework er en serie med spesifikasjoner gitt av *World Wide Web Consortium* - W3C [39], den begynte som en metadatamodell, men ble etterhvert utvidet til å kunne modellere andre typer infor-

Term Name: title	
URI:	http://purl.org/dc/elements/1.1/title
Label:	Title
Definition:	A name given to the resource.
Comment:	Typically, a Title will be a name by which the resource is formally known.
Type of Term:	element
Status:	recommended
Date Issued:	1999-07-02

Figur 3.2: Eksempel på en DCMI-term

masjon også [31]. Motivasjonen bak RDF er å kunne spesifisere metadata på en måte som gjør den aksesserbar for datamaskiner. Ulike typer media har kontekstuelle metadata assosiert med seg selv som mennesker har mye lettere for å utlede utifra kontekst enn maskiner. Det er dette problemet RDF prøver å bøte på.

For å representere metadata bruker RDF noe som kalles *statements*, hver statement inneholder en trippel som består av *objekt-predikat/verbal-subjekt*, hvert elemement i et uttrykk er identifisert med en URI, så et uttrykk kan se ut som:
(“http://uit.no/”, ”http://purl.org/dc/elements/creator”, “UiT”), som blir forstått som “`http://uit.no/ ble laget av UiT`”. Her bruker man et predikat definert av Dublin Core for gi en entydig mening til begrepet *creator*.

3.2 Trådløs

Det finnes forskjellige trådløse teknologier som man kan bruke til lokalisering, eksempler er Wi-Fi, blåtann, infrarød, ultralyd og andre. GSM hører også til her, se eget punkt under. Fellesnevneren for disse teknologiene er at de har faste aksesspunkter eller radiomaster spredt rundt i terrenget, også kalt *beacons* . Wi-Fi aksesspunkter er kanskje de mest aktuelle å bruke til posisjoneringstjenester på grunn av den voldsomme veksten i trådløse nettverk de siste årene. Men enhver radiokilde kan i teorien brukes. En av ulempene med å bruke radiokilder er at man må lage et kart over dem før man kan begynne å regne ut posisjoner. Disse må som regel byggest opp manuelt, ofte med bruk av GPS. Dette er en tidkrevende jobb og siden kartet må oppdateres jevnlig for ikke å bli for unøyaktig er det ønskelig å finne metoder som gjør denne jobben mest mulig automatisk [25].

3.3 GSM

Tjenester har beynt å dukke opp som lar brukere få tilsendt sin posisjon basert på hvilke GSM-aksesspunkter man er koblet til og avstanden til disse. Presisjonen kan være veldig variabel fra noen titalls meter til flere kilometer, avhengig av antallet basestasjoner i nærheten og kjennskap til hvor langt unna man kan plukke opp basestasjonenes signal.

Studier foreligger som viser at GSM-lokalisering kan vise til så bra presisjon som mellom 5 - 75 meter (median-verdier) [38]. Argumentasjonen for å bruke GSM istedet for GPS for lokaliseringstjenester er at GPS ikke fungerer godt der hvor folk oppholder seg, det vil si i byer og innendørs.

Den store fordelen med bruk av GSM-posisjonering er at den benytter seg av et apparat som er svært utbredt allerede, nemlig mobiltelefonen. Noe som kan være en ulempe med GSM er mangelen på basestasjoner der det ikke bor mennesker, det er ikke sannsynlig at du vil kunne få gode posisjoner hvis du er på vidda og tar bilder av et fjellvann for eksempel.

3.4 GPS

Den mest utbredte lokaliserings-teknologien i dag. Har god presisjon, informasjonssiden til den amerikanske regjeringen om GPS gir presisjonen som 100 meter i det horisontale planet og 156 meter i det verikale planet, dette er hva 95% av posisjonene vil komme innenfor, så opplevd presisjon vil som oftest være bedre enn disse verdiene, typisk presisjon er rundt 15 meter [15]. Fordelene med GPS er at teknologien virker over hele verden og at den kan gi vertikal så vel som horisontal posisjon. Ulempene er at det finnes en del steder der den ikke virker så som innendørs og steder der horisonten er høy, for eksempel i en dyp dal eller på et sted med høye bygninger. En annen faktor man må regne med ved bruk av GPS er at presisjonen til signalet er politisk regulert av den Amerikanske regjering [16]. Men den kanskje største ulempen ved bruk av GPS er strømforbruket, siden GPS må kommunisere med satelitter må de også bruke et radiosignal som er kraftig nok til å nå ut til dem. Dette krever en del strøm og for håndholdte mottakere kan dette være et problem.

3.5 Geografiske oppslagsverk

Geografiske oppslagsverk eller *Gazetteers* er gjerne alfabetiske lister som kan inneholde forskjellige opplysninger om land og steder, innbyggertall, veinummer, gatenavn og så videre, men det som er mest interessant i denne sammenhengen er at de som regel inneholder opplysninger om breddegrad og lengdegrad for stedene de lister. En del slike oppslagsverk har dukket opp på internett. Disse kan være nyttige for å knytte stedsnavn mot koordinater,

men de fleste av disse oppslagsverkene inneholder mest informasjon om land, byer og kjente landemerker. Hvis man prøver å gjøre noe på et lokalt nivå, det vil si av typen bydeler, mindre plasser, bygningsnavn og så videre er det kanskje andre alternativer man bør utforske.

Kapittel 4

Fra koordinater til stedsnavn

Dette kapittelet vil se på hvordan vi kan trekke ut informasjon om stedsnavn fra en posisjon og hvordan et system som kan dra nytte av denne informasjonen kan lages. Dette kapittelet vil primært se på teknikker som er gyldige for et system som opererer med georefererte bilder, men problemstillingene som er diskutert her kan ha relevans utenfor dette området også.

4.1 Problemstillinger

For å lage et effektivt system for å dra nytte av stedsinformasjon er det noen sentrale problemstillinger som må overvinnes. Noen av de viktigste er listet her:

- Datainnsamling
- Representasjon av geografiske lokasjoner
- Tilordne geografisk lokasjon til sted
- Bruke sted til å hente inn metadata
- Organisere og lagre metadata

4.1.1 Innsamling av geodata

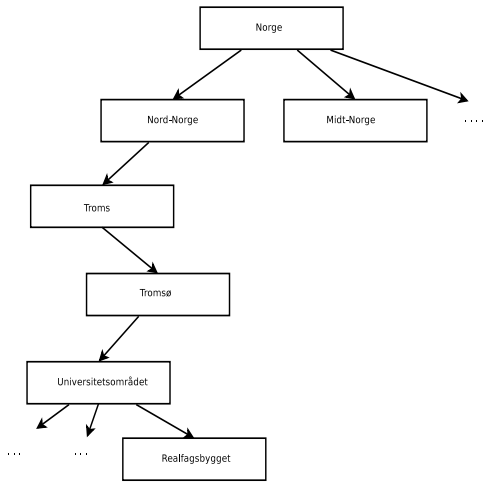
For å ha et system som drar nytte av lokalisering er man nødt til å ha informasjon om området som systemet skal omfatte. Denne informasjonen er nødt til å bli samlet inn på et tidspunkt. Dette kan være en langsom og tidkrevende prosess avhengig av størrelsen på området og kravene til systemet. Kommersielle datasett er ofte en mulighet, men da begrenser man seg til den graden av presisjon som produsenten har bestemt, dette kan være en begrensing, men igjen, det avhenger av systemet man ønsker å lage. Det trenger å nevnes at det finnes noen alternativer til kommersielle datasett

og det å samle inn dataene selv. Tilgjengelig på internett ligger det en god del geografisk informasjon mer eller mindre fritt tilgjengelig, gazetteers eller geografiske oppslagsverk kan være en god kilde for geografiske data, andre nett-tjenester er for eksempel kartbaserte tjenester slik som *google earth* [13], *google maps* [14] og kartsøk på *gulesider.no* [18]. Hvis man ikke ønsker å betale for et datasett eller trenger informasjon som ikke er tilgjengelig andre steder er alternativet å samle inn den geografiske informasjonen selv.

I sin enkleste form er datasettet bare en tekstfil som inneholder par med posisjon og navn som man kan gjøre oppslag i, men for de fleste applikasjoner er dette utilstrekkelig og effektiviteten for dårlig. Løsningen er i de fleste tilfeller en geodatabase som inneholder data om stedene og posisjon. Fordelene med en database er struktur, effektivitet ved *queries* og at den kan lagre mange andre attributter som kan være knyttet til posisjonen.

4.1.2 Representasjon av geografiske lokasjoner

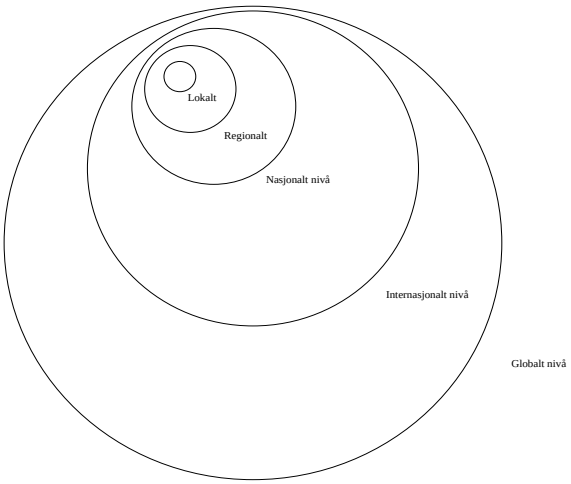
Lokasjoner trenger å være representert i systemet på et eller annet vis. Dette kan være i form av kart eller navn knyttet til posisjoner. Menneskets oppfatning av sted egner seg godt for en hierarkisk struktur, hvor man går fra store navngitte regioner til mindre navngitte regioner/plasser. Et eksempel kan bli sett i figur 4.1. Man kan se på dette som en abstraksjon av stedskonseptet. Fordelen med slike abstraksjoner er at de kan være letter å lese og forstå for mennesker, og at en slik symbolsk representasjon av data er lettere å lagre i en datastruktur.



Figur 4.1: Steder organisert i en tre-struktur

En annen måte å visualisere dette på er ved å bruke geografiske nivåer (figur 4.2). Dette kan sees på som et tre som er projisert ned på en to-

dimensjonal overflate og er den metoden som er brukt i prototypen i kapittel 5. Med en slik flat figur nærmer man seg et kart, men istedenfor en visuell representasjon av geodata har man regioner som er representert med posisjoner.



Figur 4.2: Oppdeling i nivåer

4.1.3 Tilordne geografisk lokasjon til sted

Siden geometriske koordinater ikke er de letteste forholde seg til for et menneske trenger dette systemet en mekanisme som gjør posisjoner om til stedsnavn. Her kommer datainnsamlingen man gjorde tidligere til nytte, da man trenger et geografisk datasett som tilordner posisjoner til steder. Enten man bruker absolutte eller relative posisjoneringsmetoder må man ha funksjonalitet for å gjøre posisjonene om til steder, men det er også andre faktorere man bør ha i tankene på dette punktet. Georefererte bilder inneholder en posisjonsangivelse, som regel vil ikke ikke denne posisjonen ha et nøyaktig motstykke innenfor datasettet. Det er da viktig å kunne avgjøre hvor posisjonene befinner seg i forhold til stedene som er angitt i datasettet. Man vil vite om man på dette punktet var nærmere et sted enn et annet sted, hvor langt unna man var, og i hvilken retning., man vil ha så informasjon som man kan hente ut automatisk.

4.1.4 Bruke sted til å hente inn metadata

Det neste hinderet et slikt system må overkomme er å bruke informasjonen om sted til å hente inn mer informasjon som skal kunne si noe om bildet. Men selv bare ved å hente ut et stedsnavn har man hentet ut informasjon som

er mer semantisk signifikant (for mennesker) enn et breddegrad, lengdegrad par.

Det finnes en hel del med kontekstuell informasjon man kan hente ut når man har georefererte bilder, men hvor skal man hente denne informasjonen fra? Noe kan man la systemet regne ut selv, det finnes algoritmer for å finne ut hvordan lysforholdene var på den bredde og lengdegraden på det tidspunktet som bildet ble tatt. Til andre data ting trenger man kanskje informasjon som kan hentes ut fra nettet. Vær og temperaturdata kan man finne på internett, likeså geografiske formasjoner, slik som elver, innsjøer, fjelltopper og lignende ved å bruke Gazetteers. Kanskje er kartutsnitt nyttig informasjon til et bilde og da er tjenester som Google Earth og Maps greie hjelpemidler. Hvis man kjører et system som trenger data på et lokalt nivå er det rimelig å anta at man trenger å bruke lokale ressurser, data som finnes tilgjengelig på intranett eller lignende.

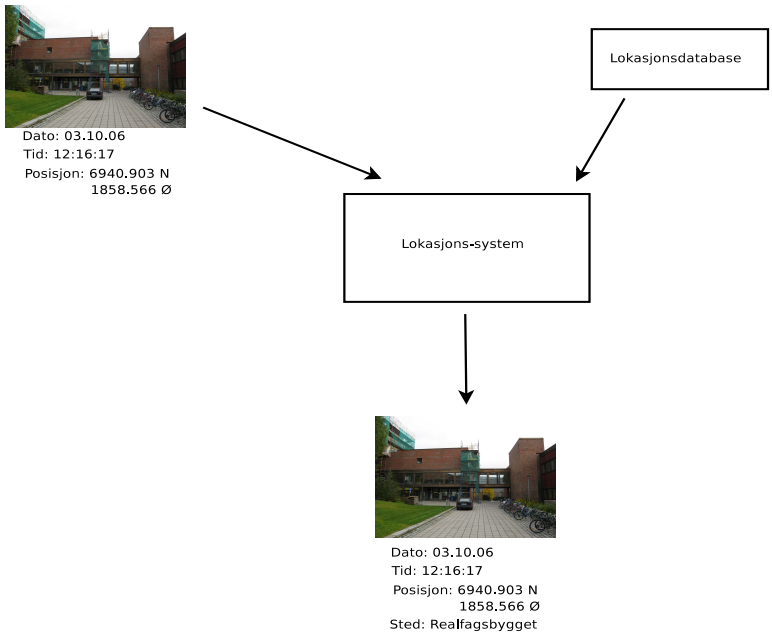
4.1.5 Organisere og lagre metadata

Informasjonen som hentes ned blir tilknyttet bildene som metadata og disse dataene må organiseres og lagres på en måte som gjør at de kan anvendes for innhenting av bilder. Her kommer metadataformater slik som Exif og RDF inn bildet. Grunnen til at vi velger å bruke slike formater til å beskrive metadata er de gjør det raskere og mere effektiv å søke etter, filtrere og identifisere innhold i bildene. Metadata vil gjerne lagres i en database som holder rede på relasjonene mellom bildene og de metadata som er hentet inn og assosiert med det.

4.2 System

Det man ønsker å sitte igjen etter å ha løst de nevnte problemstillingene er et system som har en samling av geografiske data på en eller annen form hvor en posisjon er koblet mot en lokasjon. Dette systemet skal da kunne ta inn georefererte bilder og koble dem opp mot et sted. Informasjonen om sted som nå er knyttet til bildet skal nå kunne brukes til å hente inn mer informasjon om stedet bildet er tatt fra eller objekter i bildet. Til sist trenger denne informasjonen å bli kategorisert og lagret til bildet, slik at denne informasjonen kan bli brukt senere for *image retrieval*.

I prototypen som er beskrevet i kapittel 5 er det lagt vekt på hvordan man kan gjøre posisjoner om til stedsnavn som kan lett leses av mennesker og lettere kan bli brukt for tekstbasert søk.



Figur 4.3: Konseptskisse av hva et lokasjons-system er tenkt å gjøre

Kapittel 5

Prototype

Dette kapittelet beskriver en prototype på et elektronisk oppslagsverk over bygningene på universitetsområdet i Tromsø. Prototypen er laget for å utforske problemstillinger ved det å knytte stedsnavn til posisjoner på et lokalt nivå.

Prototypen anvender seg av GPS for å angi posisjonen til bygninger. Det mest trivielle brukseksempelet er at en bruker har en GPS-posisjon som man gir til programmet, programmet vil da finne ut hvilken bygning man sto ved når GPS-målingen ble tatt, gitt at man har data om området.

5.1 Design

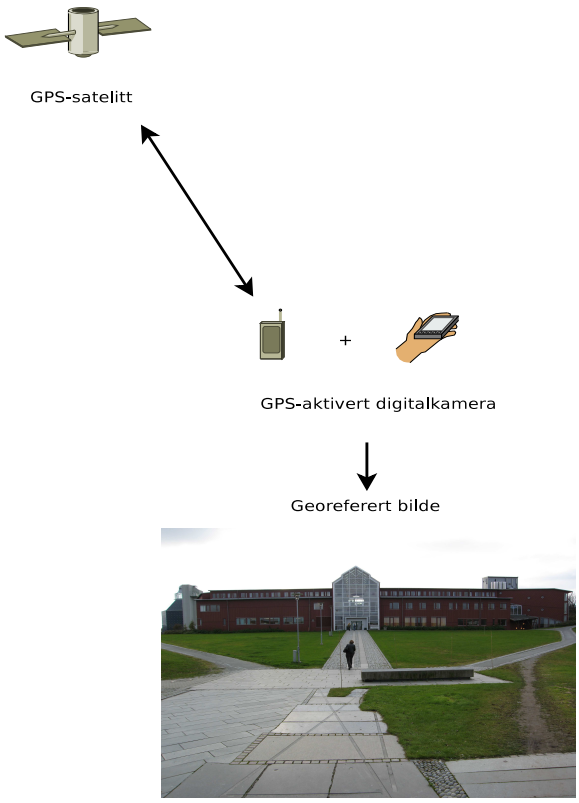
Prototypen er ment å være en *Proof of concept* av et lokasjons-system (se fig. 4.3), der man har en posisjon i form av GPS-koordinater og derifra skal man kunne gjøre oppslag i et datasett og gi tilbakemelding om hvor denne posisjonen er med et stedsnavn, i dette tilfellet med navnet på bygninger.

5.1.1 Et lite datasett

Et geografisk datasett er en samling av geografiske data som definerer et avgrenset område.

Datasettet som det ble arbeidet på her er et lite datasett med posisjonen til de fleste av Universitetet i Tromsøs sine bygninger på universitetsområdet. Det var hensiktsmessig å bruke disse siden de fleste bygningene er på et avgrenset geografisk område, det tok derfor ikke mye tid å samle inn dataene. En vanlig kommersielt tilgjengelig GPS-mottaker ble brukt for å samle inn koordinatene til byggene, hvert bygg fikk fire koordinater, en for hvert hjørne, nordøst, nordvest, sørvest og sørøst. Hvert bygg blir da representert i datasettet av et irregulært rektangel. Disse dataene er lagret i en tekstfil som er lett å parse for programmet. Posisjonene ble samlet inn med en Magellan GPS-mottaker og bruker dermed formatet for disse.

I tillegg til dette datasettet ble det laget en bildesamling av noen av byggene på universitetsområdet, disse bildene ble tatt samtidig som en sporlogg ble tatt opp av en GPS-mottaker, så man har en mulighet for å gjøre disse bildene til georefererte bilder. Denne avpasningen blir gjort ved å sammenligne dato og tidsstemplene til bildet med dato og tidsstemplene i sporloggen, så hvis ett eller flere bilder har omtrent samme tidsstempel som en oppføring i sporloggen så får de den posisjonen som ble tatt av GPS-mottakeren på det tidspunktet.



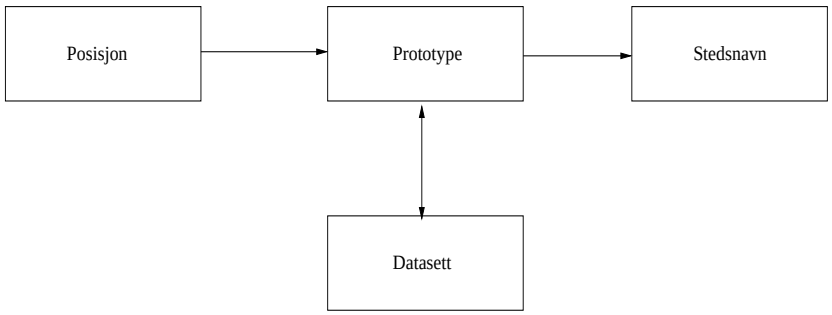
Figur 5.1: Georeferert bilde

Denne avpasningen mellom bilder og posisjon ble gjort manuelt, en overkommelige oppgave siden det bare var snakk om 40 bilder i bildesamlingen. Dette er noe som fordel kunne vært gjort automatisk.

5.1.2 Oversikt

Programmet skal kunne kjøre i 2 forskjellige moduser, den ene tar inn en enkelt GPS-posisjon og sammenligner den mot dataene i det geografiske datasettet for å finne ut om det er et navngitt sted som passer med den posisjonen. Programmet forteller da om det er et treff i datasettet, om posi-

sjonen er innenfor det definerte området men det finner ikke en direkte treff eller om posisjonen ikke er innenfor området. Den andre modusen tar inn en sporlogg fra et GPS-apparat og sjekker alle posisjons-oppføringene der mot datasettet.



Figur 5.2: Oversiktsdiagram over prototype

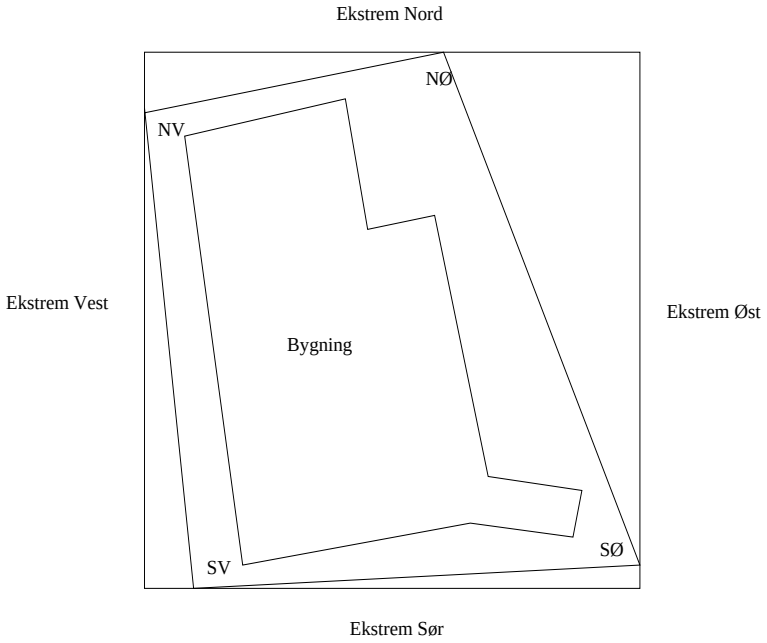
Figur 5.2 viser en oversikt over programmets oppbygning, der den tar inn en posisjon og matcher denne mot bygningene/stedene i datasettet og man får enten ut et stedsnavn eller en beskjed om at man ikke har data for den gitte posisjonen.

Prototypen anvender en ganske triviell metode for å finne ut om en posisjon er innenfor området definert i datasettet og for å finne ut om den er i nærheten av en bygning. Hele området er definert av fire koordinater som til sammen danner et stort irregulært rektangel. Programmet finner så ut ekstremverdiene for de fire himmelretningene som så danner yttergrensen for området, for et eksempel se figur 5.3.

Ved å bruke ekstremverdiene til bygget blir det omskrevet av et regulært rektangel som man så kan bruke til å beregne om en posisjon er innenfor eller ikke. Denne metoden anvendes både for området i datasettet og for bygningene.

5.2 Implementasjon

Implementasjonen av prototypen er gjort i Python på en GNU/Linux platt-form. Programmet kan kjøre med to forskjellige opsjoner, det tar enten inn et punkt i form av GPS-koordinater gitt på et spesifikt format eller en sporlogg-fil som man kan laste ned fra et GPS-apparat. Med en sporlogg



Figur 5.3: Denne figuren viser hvordan bygningene blir representert i prototypen

får man også med dato og klokkeslett i UTC for når posisjonen ble tatt. Se vedlegg A.1 for kildekode.

Det første som skjer er at programmet bygger opp oppslagsverket over campus, dette skjer ved at det leser inn en tekstfil over området som skal legges inn. Denne fila inneholder data over bygningene på området og organisert slik at hvert bygg har et unikt navn og til hvert bygg er det fire GPS-posisjoner som angir hjørnene nordøst, nordvest, sørvest og sørøst. Denne fila finnes i vedlegg A.2. Disse dataene blir så lagt inn i et python-dictionary, kalt *map_dict*, der hvert bygg blir representer med et nøkkel-verdi par. Nøkkelen er et nummer fra 1 til antallet bygg som ligger i kartfila, mens verdien er en liste-struktur som inneholder navnet på bygget samt posisjonene til de 4 hjørnene pluss en del data som GPS-mottakeren lagrer men som ikke brukes i programmet. En vanlig oppføring i *map_dict* kan se slik ut:

```
['# FARMASI\n', ['NE $PMGNWPL', '6941.004', 'N', '01858.641', 'E',
 '0000287', 'M', 'POI001', '', 'a*37\n'], ['NW $PMGNWPL', '6940.968',
 'N', '01858.538', 'E', '0000035', 'M', 'POI001', '', 'a*33\n'],
['SW $PMGNWPL', '6940.944', 'N', '01858.495', 'E', '0000030', 'M',
'POI001', '', 'a*3E\n'], ['SE $PMGNWPL', '6940.919', 'N',
'01858.581', 'E', '0000026', 'M', 'POI001', '', 'a*35\n']]
```

Man ser at hvert hjørne, gitt ved NE, NW, SW og SW har nordlig

breddegrad og østlig lengdegrad. Ut i fra denne oppføringen kan man lese at det nordøstre (NE) hjørnet til Farmasi-bygget ligger på 69.41 ° og 0.04 minutter nordlig bredde og 18.581 ° 6.41 minutter østlig lengde.

For å avgjøre om en gitt posisjon er innenfor det definerte området henter programmet ut de enkelte GPS-posisjonene til en bygning og henter ut de ekstreme nord, sør, øst og vest verdiene, dette gjør det enkelt å bestemme om en posisjon er innenfor området avgrenset av de koordinatene. Posisjonen man får inn har en en nordlig breddegrad og en østlig lengdegrad man må da sjekke om disse ligger i mellom grensene til området satt tidligere. Denne metoden brukes også for å avgjøre om en posisjon er i nærheten av en bygning.

5.2.1 Formater

En kort forklaring til formatene som er brukt i denne oppgaven. Tekstfilen som inneholder posisjonsdataene til bygningene er bygd opp av posisjoner hentet inn med en Magellan Explorist 500 GPS-mottaker[17], og så organisert og lagt inn i en fil av meg. Her er et utdrag fra fila for Farmasi-bygget:

```
# FARMASI
NE $PMGNWPL,6941.004,N,01858.641,E,0000287,M,POI001,,a*37
NW $PMGNWPL,6940.968,N,01858.538,E,0000035,M,POI001,,a*33
SW $PMGNWPL,6940.944,N,01858.495,E,0000030,M,POI001,,a*3E
SE $PMGNWPL,6940.919,N,01858.581,E,0000026,M,POI001,,a*35
```

FARMASI er navnet på bygget, og angir at her kommer 4 nye posisjoner for hjørnene på området som inneholder bygningen. Hver posisjon er prefikset med den generelle himmelretningen, dette er gjort for å gjøre fila lettere å lese og for debugging. Det neste som følger er selve posisjonsangivelsen som den blir lagret av GPS-en. For en forklaring av formatene se Vedlegg A.2 og A.3

5.3 Testing og resultater

Tester ble kjørt med to sporlogger som inneholdt posijoner både på og utenfor universitetsområdet. Den ene av disse sporloggene var tatt opp samtidig som en bildesamling med bilder både på og utenfor universitetsområdet. De georeferert bildene kunne da brukes til å avgjøre om algoritmen for å finne ut om man var i nærheten av et bygg fungerte som den skulle. De avdekket også noen problemer med bruk av en regulær GPS-mottaker i samband med et digitalkamera. Se Diskusjon.

Her er resultatet fra kjøringen av programmet med en sporlogg som opsjon:

```
kaiab@kaiab:~/diplom/kode$ ./campus.py -t T1.log
```

```
Den 031006 klokken 120506.60 var du ved # NFH

Den 031006 klokken 120620.14 var du ved # TEORIFAG-VEST

Den 031006 klokken 120726.59 var du ved # TEORIFAG-ØST

Den 031006 klokken 120758.60 var du ved # TEORIFAG-ØST

Den 031006 klokken 121211.61 var du ved # ISV

Den 031006 klokken 121249.61 var du ved # ISV

Den 031006 klokken 121412.60 var du ved # ISV

Den 031006 klokken 121617.56 var du ved # MAT-NAT

Slutt på sporlogg, avslutter...$
```

Etter kjøringen av programmet med test-sporloggen `T1.log` (se vedlegg A.3) som opsjon kom den opp med 8 treff for bygninger som programmet hadde klart å avgjøre at jeg hadde vært i nærheten av. For å teste ut dette sjekket jeg tidspunktene for disse treffene opp mot tidspunkt da bildene ble tatt. På denne måten kunne jeg sjekke om et bilde tatt på omtrent samme tidspunkt som posisjonen i sporloggen virkelig var i nærheten av bygningen. Jeg fant 10 bilder i bildesamlingen som alle hadde blitt tatt på et tidspunkt som matchet med tidspunkt.

Når vi så begynner å finne bilder i bildesamlingen som ble tatt samtidig med sporloggen for å finne bilder som matcher tidspunktet dukker det opp noen problemer. Hvis vi ser på bildet i 5.4 som ble tatt klokken 14.06 og kobler det sammen med det nærmeste tidspunktet i sporloggen, finner vi at klokken 14:06:20 (12:06:20 UTC+2 timer) har GPS-mottakeren registrert posisjonen min som 69.40.899 N og 18.58.286 E og den skal ligge ved den vestlige delen av Teorifagbygget. Problemet er at bildet ble tatt like ved NFH som ligger omtrent 100 meter lenger sør på universitetsområdet. Her ser vi et av problemene ved å bruke en GPS-sporlogg på et så lokalt nivå, små feil blir store, i dette tilfellet er ikke klokkene på GPS-en og digitalkameraet synkronisert, i tillegg blir ikke posisjonene tatt i samme øyeblikk som bildene så det vil bli litt ulike tidspunkt, noe man må ta høyde for i slike applikasjoner.

5.4 Begrensinger

På grunn av høyt tidspress var det nødvendig å sette noen begrensinger på prototypen. Koden er veldig lokalisert og fungerer bare på koordinater tatt

Figur 5.4: Bilde av hovedinngangen til Norges Fiskerihøyskole

på den nordlige og østlige halvkule, den vil ikke fungere med koordinater med negativt fortegn. Prototypen anvender også et bestemt format på koordinatene, gitt av produsenten av GPS-mottakeren som ble anvendt. Andre GPS-mottakere kan ha andre formater av koordinatene så konvertering mellom forskjellige formater er en god ide.

Kapittel 6

Diskusjon

Her følger en diskusjon og kritikk av det arbeidet som ble gjort med prototypen. Vi ser også på problemer ved å arbeide så lokalt og utfordringer ved å bruke GPS knyttet til dette.

6.1 Bruksområder

Et digitalt "bygningskart" over universitetsområdet som kan fortelle en bruker eller applikasjon hvor, i form av et stedsnavn, man befinner seg. For eksempel man tar et bilde av bygningen Hovedgården med et GPS-aktivert digitalkamera, kameraet lagrer posisjonen i bildets EXIF-data, senere vil man kunne hente ut denne posisjonen og bruke et program til å hente ut navnet på bygningen eller stedet. Hva man så velger å gjøre med denne informasjonen er opp til brukeren. Mulige anvendelser vil være gi bildet ny tittel eller å hente ut mer informasjon om objektet gjennom søk. Det er selfølgelig andre bruksområder for et lokasjonssystem, slik som for eksempel navigasjon, turistguide, men de faller litt utenfor fokuset for denne avhandlingen.

6.2 Kritikk og forslag til utvidelser av prototypen

Det er en rekke forbedringer og utvidelser som kan gjøres med prototypen som er beskrevet i kapittel 5, blant annet etablere en hierarkisk lokasjonsdatabase, gjøre algoritmen for posisjonering mer robust, forbedre mekanismen for å bestemme posisjon forhold til et sted, parsing av bildefiler, legge til en mekanisme som kan hente inn kontekstuell metadata koblet til sted og assosiere metadataene med bildene.

I prototypen bruker vi et lite datasett som gir oss informasjon over et begrenset område og lagerer informasjon om 14 bygninger innenfor dette området. Innenfor rammene til prototypen er dette greit å lagre i en tekstfil,

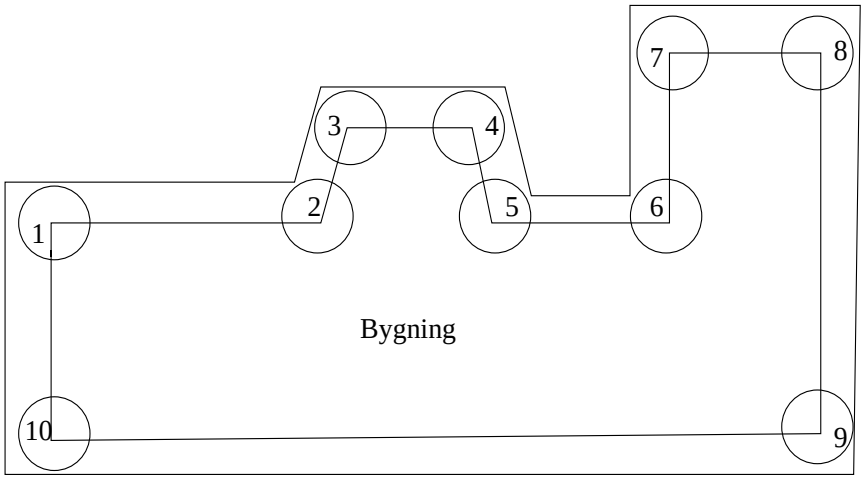
men i et system som faktisk kunne brukes for lokasjonsangivelse der man antakelig ville ha hatt data for større områder og ikke minst mange flere områder ville dette ikke fungere veldig effektivt. Så det man ønsker er en database som samler sammen posisjoner og tilhørende steder, men dette er ikke de eneste data som kan være i databasen, informasjon som ikke trenger å regnes ut i sanntid kan lagres i denne databasen, typiske elementer er geografiske særpreg, landemerker og så videre, disse er som regel statiske attributter til et sted og kan hentes ned på forhånd om det er ønskelig.

Områder i datasettet er representert ved et sett med nord, sør, øst og vest verdier som danner et rektangel, disse koordinatene utledes fra koordinatene for hver bygning som ligger i datasettet. I et større system med flere områder måtte dette gjøres når man legger inn data, slik at man samler inn data for et område og navngivingen av området følger av hvilket område som omfatter koordinatene til stedene. Prototypen har bare ett område, nemlig universitetsområdet og grensene til dette området er bestemt av de bygningene som er med. For å ta et eksempel, la oss si at man ønsket å lage et geografisk datasett for sentrum av Tromsø, da ville man under datainnsamlingen hente koordinater for hver bygning i sentrum og dette ville være et område, man kunne da når man legger inn data i en database la Storgata være et eget subområde av Sentrum ved å la de stedene/bygningene som ligger i Storgata definere dette området.

En annen problemstilling som ikke er løst fullgodt i min implementasjon er hvordan man finner ut om et punkt er i nærheten av en bygning. Som nevnt i kap. 5.2 gjøres dette nå ved å bruke et rektangel definert av nord, sør, øst og vest verdiene til et bygg og deretter avgjøre om et punkt er innenfor grensene satt av disse verdiene ved en enkel sammenligning, men dette gjør at for bygg som ikke er formet som et rektangel (de fleste viser det seg) kan regionen rundt bygget bli for stor noen steder og for snever andre steder. Et problem som kan løse dette er å la byggene bli representert av mange polygoner i stedet for firkanter, men dette vil fordre at man tar langt flere målinger av et bygg. En annen måte å gjøre dette på er å la hvert hjørne på et bygg ha en en region som for ekspel har en radius på 10 meter, når man så trekker en linje mellom regionene har man en sone rundt bygget som man bruke til å finne ut om man er i nærheten. Man trenger da en viss disiplin under datainnsamlingen siden man må ta koordinatene for hvert hjørne i en bestemt rekkefølge for å kunne bestemme hvilket hjørne som er nært et annet etterpå, se fig. 6.1.

Dette vil gi en mer nøyaktig posisjon og mindre fare for at områder rundt bygninger overlapper. En slik representasjon gir også nye utfordringer når man skal avgjøre om et georeferert bilde ble tatt innenfor denne regionen.

En annen funksjon som er ønskelig å ha i et slikt program er avstand



Figur 6.1: Alternativ representasjon av bygninger

og retning, avstand er relativt enkelt å beregne mellom 2 koordinater på overflaten av en kule og innebærer noen geomeriske beregninger. Avstand kan være nyttig ha i metadatabeskrivelsen av et bilde, et område vil som oftest bestå av udefinert rom mellom kjente steder og landemerker og det er da ønskelig å kunne si noe om hvor man er i forhold til de kjente stedene. Så det skal være mulig å tagge et bilde med for eksempel "Universitetsområdet i Tromsø, bildet tatt 50 meter sør for Labyrinten". Som gir en pekepinn om hvor man er i verden og inneholder en relasjon til et landemerke.

6.2.1 Objekter i bildet

Et av spørsmålene som kom opp under arbeidet med denne avhandlingen var om et slikt lokasjonssystem kan hjelpe til med identifisering av objekter i bildet? Etter min mening er svaret ja, med forbehold. For å identifisere objekter i bildet er det en del forutsetninger som bør være på plass. Apparatet som man tar bildet med må være GPS-aktivert eller på annen måte ha en mulighet for å lage georefererte bilder, i tillegg er det ønskelig å vite i hvilken retning man har kameraet rettet mot, så man trenger et kompass, videre kan det nyttig å vite avstanden, avstanden til objektet kan enten regnes ut basert på de posisjonene til objektet og hvor man sto når man tok bildet eller man kan bruke en avstandsmåler som lagrer avstanden sammen med de andre dataene. Dette skulle være nok informasjon til at man i de aller fleste tilfeller kan klare å identifisere et landemerke/bygning/sted som man tar bilde av, hvis det finnes datasett som man kan gjøre oppslag i som har informasjon om objektet. Denne fremgangsmåten fungerer selvfølgelig bare for objekter som kan identifiseres ved hjelp av lokasjon.

For prototypen viste det seg at det var vanskelig å anvende bare GPS-posisjonen for å avgjøre om en bygning var i bildet. Sporloggen lagrer ikke retning, selv om det er mulig å ekstrapolere en generell retning utifra posisjonene i sporloggen. Men siden man som regel ser og beveger seg rundt for å finne den beste vinkelen eller motivet før man tar bildet er ikke denne informasjonen nok.

6.2.2 Problemer/Utfordringer ved bruk av GPS

Under arbeidet med prototypen og innsamlingen av geodata kom opp en del problemer ved bruk av GPS-mottakeren. Et kjent problem er selvfølgelig at den ikke fungerer spesielt godt innendørs og mellom høye bygningen, GPS-en rapporterte ved et tilfelle klemt inn mellom to høye bygninger at den hadde en presisjon på 240 meter, noe som for det meste er greit ute i terrenget, men når man prøver å få relativt nøyaktige posisjoner for stedsangivelse er dette et problem, som regel rapporterte mottakeren at den hadde en presisjon på mellom 4 og 8 meter, noe som er akseptabelt i denne sammenhengen.

Videre viste det seg at å bruke sporlogg-funksjonen til en håndholdt GPS-mottaker istedenfor en som er integrert i kameraet som tar en posisjon når bildet blir tatt og lagrer denne i samband med bildet viste seg å by på problemer når man prøver en så lokal posisjonering. For det første må kameraets klokke stilles inn etter GPS-mottakerens klokke, slik at tidspunktene er mest stemmer overens, men selv om de er synkronisert perfekt er det mulig gå et godt stykke på den tiden det tar GPS-en å oppdatere en sporlogg. Dette kan gjøre at man bildet blir feilaktig posisjonert og dermed at stedsangivelsen blir feil. Som regel er ikke dette et stort problem og med en god algoritme som matcher en sporlogg med en bildesaming kan dette problemet bli til dels lindret.

6.3 Lokal og regional posisjonering

I denne oppgaven valgte jeg et litt annet fokus enn andre prosjekter som for eksempel `NameSet` (se kap. 2.6.4). `NameSet` gjør posisjonering på et høyere nivå enn det som er forsøkt studert her. Objektene som man bruker som referanser i deres prosjekt er som regel flere kilometer eller mil unna posisjonen eller så er posisjoneringen regional i stedet for lokal. Eksempler er at de bruker byer eller nasjonalparker som referanser, mens dette kanskje er tilstrekkelig for å fungere som et hjelpemiddel for browsing gjennom en bildesamling og for å samle inn metadata, vil vi argumentere for at man kan samle inn mer relevante data gjennom et system som har lokale referanser og at mulig identifisering av objekter i bildet er mer sannsynlig. Men dette er avhengig av datasettet som er tilgjengelig.

6.4 Lokasjon som utgangspunkt for innhenting av informasjon

Etter alt snakket om lokasjon er det viktig å ikke miste av synet hvorfor man vil ha denne informasjonen på plass. Det er et virkemiddel for å kunne hente inn metadata til bildet som kan hjelpe brukere til å søke/browse etter bilder på nettet og i store bildesamlinger.

Kapittel 7

Konklusjon

Dette kappitelet vil avslutte og konkludere denne avhandlingen, her vil vi se på hvilke konklusjoner vi kan dra av arbeidet med prototypen og vi vil også prøve si litt om hvilken retning vi ser denne typen vil ta i fremtiden.

Denne avhandlingen har diskutert måter å knytte stedsnavn til bilder ved bruk av teknologier for posisjonering og måter å anvende denne informasjonen til å generere metadata ved å bruke kontekst som er betinget av lokasjon. Avhandlingen har spesielt hatt som fokus metoder for å oversette posisjoner i form av koordinater til stedsnavn. Det ble laget en prototyp for å se hvordan GPS-lokalisering fungerer på et lokalt nivå.

Prototypen ble laget for å teste om GPS-posisjonering sammen med et sett med geografiske data over et lite begrenset område ville kunne gi gode stedsangivelser. For å teste dette konseptet ble prototypen av et lokasjonssystem testet med en liten bildesamling som ble samkjørt med en GPS-sporlogg som ble tatt samtidig som bildene. Vi fant at prototypen fungerte tilfredsstillende selv om det ble avdekket noen problemer ved å anvende GPS på dette nivået, samt noen problemer med noen av metodene brukt i prototypen. Problemene med prototypen var av teknisk art og gikk på at metodene for posisjonering og bestemming av regioner rundt bygninger ikke var robuste og presise nok.

7.1 Evaluering av målsetning

Målet for denne avhandlingen var å undersøke om hvordan man ved å bruke koordinater kunne gjøre dem om til stedsnavn som er forståelig for mennesker og anvende disse til *image retrieval*, underveis ble det også klart at man ville undersøke dette i en lokal setting. Til dette målet konkluderer vi med at GPS-apparater håndholdte eller integrert med kameraet er gode hjelpe-

midler for å lage georefererte bilder og med en fungerende om enn primitiv prototype som gir disse posisjonene et stedsnavn vil vi argumentere for at målet er delvis oppfylt. Det er fortsatt en del åpne spørsmål når det gjelder å anvende dette innenfor innhenting av bilder, blant annet gjelder dette hvordan man best kan hente inn metadata som er knyttet til lokasjon og organisere metadata assosiert med et bilde på en måte som gjør det lettere for en bruker å søke etter bilder.

7.2 Fremtidig arbeid

Det er fortsatt en god del arbeid som kan gjøres på dette feltet. Innhenting av bilder basert på stedlig kontekstinformasjon kan minske det semantiske skillet, men de kan ikke løse problemet. Det ville vært interessant å kombinere teknikker for innholdsbasert *information retrieval* med lokasjons og kontekstbaserte metoder for å se om man kunne øke andelen relevante resultater fra bildesøk.

GPS har noen begrensinger på et så lokalt nivå som er beskrevet i denne avhandlingen, i tettbebygde strøk kan det være lurere å benytte seg av trådløse lokasjonsteknologier slik som GSM og utstyr i 802.11 familien. Her blir den store fordelen med GPS at den er globalt tilgjengelig ikke lengre så viktig gitt den store utbyggingen av mobilnettet og trådløse nettverk.

Også innenfor metadata og hvordan men velger å hente inn, organisere og lagre disse er det mye arbeid igjen. Automatisk annotering av bilder med metadata er en nødvendighet for at *image retrieval* fra nettet og fra store bildesamlinger/databaser skal bli virkelig effektivt.

Bibliografi

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] Alipr hjemmeside. http://www.alipr.com/.

[3] Christian Becker and Frank Dürr. On location models for ubiquitous computing. *Personal Ubiquitous Comput.*, 9(1):20–31, 2005.

[4] P.J. Brown and G.J.F. Jones. Context-aware retrieval: Exploring a new environment for information retreival and information filtering, 2001.

[5] Sun or moon rise/set table for one year. http://aa.usno.navy.mil/data/docs/RS_OneYear.html.

[6] Dublin core metadata initiative. http://dublincore.org/.

[7] Dublin core metadata terms. http://dublincore.org/documents/dcmi-terms/.

[8] eklima. http://eklima.no.

[9] Exif unofficial homepage. http://exif.org/.

[10] Exif standard. http://www.digicamsoft.com/exif22/exif22/html/exif22_1.htm.

[11] Flickr homepage. http://www.flickr.com/.

[12] Google homepage. http://www.google.com/.

[13] Google earth. http://earth.google.com/.

[14] Google maps. http://maps.google.com/.

[15] Official us gps information homepage. http://gps.gov/.

[16] Us gps policy. http://pnt.gov/policy/.

[17] Magellan explorist 500 manual. http://www.magellangps.com/assets/manuals/eXplorist_50

[18] Gule sider kartsøk. http://kart.gulesider.no/kart/index.c.

[19] Jonathon S. Hare, Patrick A.S. Sinclair, Paul H. Lewis, Kirk Martinez, Peter G.B. Enser, and Christine J. Sandom. Bridging the semantic gap in multimedia information retrieval: Top-down and bottom-up approaches. In *Proceedings of Mastering the Gap: From Information Extraction to Semantic Representation / 3rd European Semantic Web Conference*, 2006.

[20] Masashi Inoue. On the need for annotation-based image retrieval. In *Workshop on Information Retrieval in Context*, pages 44–46, 2004.

[21] Masashi Inoue and Naonori Ueda. Retrieving lightly annotated images using image similarities. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1031–1037, New York, NY, USA, 2005. ACM Press.

[22] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(3):58–68, 2005.

[23] M. Kherfi, D. Ziou, and A. Bernardini. Atlas wise: A web-based image retrieval engine. In *Proceedings of the International Conference on Image and Signal Processing (ICISP Agadir)*, pages 69–77, 2003.

[24] M. L. Kherfi, D. Ziou, and A. Bernardi. Image retrieval from the world wide web: Issues, techniques, and systems. *ACM Comput. Surv.*, 36(1):35–67, 2004.

[25] Anthony LaMarca, Jeffrey Hightower, Ian Smith, and Sunny Consolvo. Self-mapping in 802.11 location systems. In *Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp 2005)*, Lecture Notes in Computer Science, pages 87–104. Springer-Verlag, September 2005.

[26] Jia Li and James Z. Wang. Real-time computerized annotation of pictures. In *Proceedings of the ACM Multimedia Conference*, pages 911–920, 2006.

[27] Mappr! homepage. http://www.mappr.com/.

[28] Mor Naaman, Susumu Harada, QianYing Wang, Hector Garcia-Molina, and Andreas Paepcke. Context data in geo-referenced digital photo collections. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 196–203, New York, NY, USA, 2004. ACM Press.

[29] Mor Naaman, Yee Jiun Song, Andreas Paepcke, and Hector Garcia-Molina. Automatic organization for digital photographs with geographic coordinates. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 53–62, New York, NY, USA, 2004. ACM Press.

[30] Mor Naaman, Yee Jiun Song, Andreas Paepckea, and Hector Garcia-Molina. Assigning textual names to sets of geographic coordinates. volume 30, pages 418–435. Elsevier, 2006.

[31] Rdf specifications. http://www.w3.org/RDF/.

[32] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues, April 1999.

[33] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[34] Bill N. Schilit, Anthony LaMarca, Gaetano Borriello, William G. Griswold, David McDonald, Edward Lazowska, Anand Balachandran, Jason Hong, and Vaughn Iverson. Challenge: ubiquitous location-aware computing and the place lab"initiative. In *WMASH '03: Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 29–35, New York, NY, USA, 2003. ACM Press.

[35] S. Sclaroff, L. Taycher, and M. La Cascia. Imagerover: A content-based image browser for the world wide web. In *Proceedings of IEEE Workshop on Content-based Access of Image and Video Libraries*, 1997.

[36] J. R. Smith and S.-F. Chang. An image and video search engine for the world-wide web. In *Proceedings, IS&T/SPIE Symposium on Electronic Imaging: Science and Technology (EI'97) - Storage and Retrieval for Image and Video Databases V*, 1997.

[37] Kentaro Toyama, Ron Logan, and Asta Roseway. Geographic location tags on digital images. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 156–166, New York, NY, USA, 2003. ACM Press.

[38] Alex Varshavsky, Mike Chen, Eyal de Lara, Jon Froehlich, Dirk Haehnel, Jeffrey Hightower, Anthony LaMarca, Fred Potter, Timothy Sohn, Karen Tang, , and Ian Smith. Are gsm phones the solution for localization? In *IEEE Workshop on Mobile Computing Systems and Applications*, April 2006.

[39] World wide web consortium homepage. http://www.w3.org/.

[40] World geodetic system 1984. http://earth-info.nga.mil/GandG/wgs84/.

[41] Yahoo homepage. http://www.yahoo.com/.

[42] R. Zhao and W. Grosky. Bridging the semantic gap in image retrieval.

Tillegg A

Vedlegg

A.1 Kildekode

```
#!/usr/bin/python

import sys
import os
import getopt

mapfile = 'gpscampus.txt'
#mapfile = 'testfil.txt'
tracklog = 'T1.log'

def parse_mapfile(num_b):
    linec = 0
    numb = 0
    mdict = {}
    new_entry = []
    f = open(mapfile,'r')

    for line in f:
        if numb == num_b:
            break
        if line[0] == '#':
            line.strip()
            #print line, "lagt til"
            new_entry.append(line)
        elif linec == 0:
            #print "linec = 0"
            new_entry.append(make_entry(line))
            linec += 1
        elif linec == 1:
            #print "linec = 1"
            new_entry.append(make_entry(line))
            linec += 1
        elif linec == 2:
            #print "linec = 2"
            new_entry.append(make_entry(line))
            linec += 1
        elif linec == 3:
            #print "linec = 3"
            new_entry.append(make_entry(line))
            numb += 1
            mdict.__setitem__(numb,new_entry)
```

```
            new_entry = []
            linec = 0


    f.close()

    #print "numb = ", numb

    #print mdict

    return mdict



def make_entry(line):
    #print line
    entry = []

    line.strip()
    line_parts = line.split(',')
    for i in range(len(line_parts)):
        entry.append(line_parts[i])

    return entry


def num_buildings():
    buildings = 0
    f = open(mapfile, 'r')

    for line in f:
        if line[0] == '#':
            buildings += 1

    f.close()

    return buildings


# Henter ut hjørnene til hele området
def define_area(map_dict, numb):
    print "define"

    for i in range(1,numb+1):
        building = map_dict.get(i)
        name = building[0]
        ne = float(building[1][1]), float(building[1][3])
        nw = float(building[2][1]), float(building[2][3])
        sw = float(building[3][1]), float(building[3][3])
        se = float(building[4][1]), float(building[4][3])

        if i == 1:
            #print "hit"
            #former_name = name
            # North
            if ne[0] > nw[0]:
                extreme_n = ne[0]
            else:
                extreme_n = nw[0]
            # East
            if ne[1] > se[1]:
                extreme_e = ne[1]
```

```
            else:
                extreme_e = se[1]
            # South
            if sw[0] < se[0]:
                extreme_s = sw[0]
            else:
                extreme_s = se[0]
            # West
            if nw[1] < sw[1]:
                extreme_w = nw[1]
            else:
                extreme_w = sw[1]
            #print extreme_n, extreme_s, extreme_w, extreme_e
        else:
            #print "dit"
            # Find extreme north
            if max(ne[0], nw[0]) > extreme_n:
                extreme_n = max(ne[0], nw[0])
            # Find extreme west
            if  min(sw[1], nw[1]) < extreme_w:
                extreme_w = min(sw[1], nw[1])
            # Find extreme east
            if max(se[1], ne[1]) > extreme_e:
                extreme_e = max(se[1], ne[1])
            # Find extreme south
            if min(se[0], sw[0]) < extreme_s:
                extreme_s = min(se[0], sw[0])

    #print extreme_n, extreme_s, extreme_e, extreme_w
    corners_nsew = extreme_n, extreme_s, extreme_e, extreme_w

    return corners_nsew

# Determine if a coordinate is in the mapped aread
def in_area(corners, point):
    coordinate = float(point[0]), float(point[1])
    #print "n = ", coordinate[0]
    #print "s = ", coordinate[1]
    #print "Corners =", corners
    #print corners[0]
    #print corners[1]
    #print corners[2]
    #print corners[3]
    if coordinate[0] <= corners[0]
and coordinate[0] >= corners[1]
and coordinate[1] <= corners[2]
and coordinate[1] >= corners[3]:
        print "Du er på campus"
        return 1
    else:
        print "Du er ikke på campus"
        return 0


# Find which building the coordinate is closest too
def find_building(map_dict, point, numb):
    #print "find"
    coordinate = float(point[0]), float(point[1])

    #if in_area() == 0:
     #   sys.exit(0)
```

```
    for i in range(1, numb):
        building = map_dict.get(i)

        name = building[0]
        #print name

        ne = float(building[1][1]), float(building[1][3])
        nw = float(building[2][1]), float(building[2][3])
        sw = float(building[3][1]), float(building[3][3])
        se = float(building[4][1]), float(building[4][3])

        # Find extreme north
        extreme_n = max(ne[0], nw[0])
        # Find extreme west
        extreme_w = min(sw[1], nw[1])
        # Find extreme east
        extreme_e = max(se[1], ne[1])
        # Find extreme south
        extreme_s = min(se[0], sw[0])

        #print extreme_n, extreme_s, extreme_e, extreme_w
        corners = extreme_n, extreme_s, extreme_e, extreme_w

        if coordinate[0] <= corners[0]
and coordinate[0] >= corners[1]
and coordinate[1] <= corners[2]
and coordinate[1] >= corners[3]:
            #print "Du er ved ", name
            return name



def usage():
    print """Usage:

    campus.py -f <coordinate> [-t <tracklog.log>]

    Options:
            -h display this message
            -f give a coordinate on the form (latitude, longitude)
            -t optional, specify a tracklog file

    Examples:
             python campus.py -t tracklog.log
             python campus.py -f 6941.025,1858.305
    """

def parse_tracklog(tracklog, map_dict, numb):
    f = open(tracklog, 'r')

    for line in f:
        line.strip()
        parts = line.split(',')
        point = parts[1], parts[3]

        name = find_building(map_dict, point, numb)
        time = parts[7]
        date = parts[10]
        date = date.strip()
        tmpdate = date.split('*')
        date = tmpdate[0]
```

```
        if name != None:
            print "Den", date, "klokken", time, "var du ved", name



def main(argv):
    #tracklog = 'TEST.log'
    coordinate = None
    mapfile = 'gpscampus.txt'

    num_b = num_buildings()
    map_dict = parse_mapfile(num_b)
    corners = define_area(map_dict, num_b)

    try:
        opts, args = getopt.getopt(argv, "hf:t:")
    except getopt.GetoptError, error:
        print error
        usage()
        sys.exit()

    for o, a in opts:
        if o in ('-f'):
            a = a.strip()
            coordinate = a.split(',')
            if in_area(corners, coordinate):
                name = find_building(map_dict, coordinate, num_b)
                print "Du er ved ", name
            else:
                sys.exit()
        if o in ('-h'):
            usage()
            sys.exit()
        if o in ('-t'):
            tracklog = a
            parse_tracklog(tracklog, map_dict, num_b)



if __name__ == '__main__':
    main(sys.argv[1:])
```

A.2 Kartfil

Nedenfor følger en forklaring på de forskjellige feltene som GPS-en lagrer. Leses fra venstre mot høyre, uttrykket er kommaseparert:

```
$PMGNWPL,6941.004,N,01858.641,E,0000287,M,POI001,,a*37
```

1. Angir protokollen
2. Breddegrad, gitt på grader, minutter og desimalminutter
3. Angir nordlig eller sørlige breddegrad
4. Lengdegrad
5. Angir om det er østlig eller vestlig lengdegrad
6. Elevasjon
7. Angir om elevasjon er uttrykt i meter eller fot
8. Navn på punktet
9. Plass for en beskjed
10. Siste delen av formatet angir typen til punktet som kan være forskjellig mellom GPS-modeller

```
# FARMASI
NE $PMGNWPL,6941.004,N,01858.641,E,0000287,M,POI001,,a*37
NW $PMGNWPL,6940.968,N,01858.538,E,0000035,M,POI001,,a*33
SW $PMGNWPL,6940.944,N,01858.495,E,0000030,M,POI001,,a*3E
SE $PMGNWPL,6940.919,N,01858.581,E,0000026,M,POI001,,a*35
# NFH
NE $PMGNWPL,6940.991,N,01858.487,E,0000043,M,POI001,,a*31
NW $PMGNWPL,6941.011,N,01858.411,E,0000055,M,POI001,,a*39
SW $PMGNWPL,6940.957,N,01858.300,E,0000048,M,POI001,,a*38
SE $PMGNWPL,6940.939,N,01858.363,E,0000041,M,POI001,,a*3C
# MH
NE $PMGNWPL,6941.010,N,01858.808,E,0000061,M,POI001,,a*3B
NW $PMGNWPL,6941.037,N,01858.694,E,0000053,M,POI001,,a*34
SW $PMGNWPL,6940.972,N,01858.546,E,0000042,M,POI001,,a*31
SE $PMGNWPL,6940.948,N,01858.659,E,0000080,M,POI001,,a*3B
# MAT-NAT BIBLIOTEKET
NE $PMGNWPL,6940.951,N,01858.724,E,0000089,M,POI001,,a*31
NW $PMGNWPL,6940.953,N,01858.717,E,0000083,M,POI001,,a*39
SW $PMGNWPL,6940.953,N,01858.684,E,0000152,M,POI001,,a*3F
SE $PMGNWPL,6940.932,N,01858.704,E,0000066,M,POI001,,a*37
```

```
# MAT-NAT
NE $PMGNWPL,6940.926,N,01858.713,E,0000081,M,POI001,,a*3D
NW $PMGNWPL,6940.938,N,01858.662,E,0000077,M,POI001,,a*3C
SW $PMGNWPL,6940.899,N,01858.557,E,0000083,M,POI001,,a*38
SE $PMGNWPL,6940.873,N,01858.601,E,0000052,M,POI001,,a*30
# FISK-FORSK
NE $PMGNWPL,6940.923,N,01858.592,E,0000043,M,POI001,,a*3D
NW $PMGNWPL,6940.943,N,01858.518,E,0000059,M,POI001,,a*32
SW $PMGNWPL,6940.922,N,01858.454,E,0000049,M,POI001,,a*3D
SE $PMGNWPL,6940.894,N,01858.527,E,0000055,M,POI001,,a*39
# ØVRE LYSTHUS
NE $PMGNWPL,6940.884,N,01858.510,E,0000046,M,POI001,,a*3E
NW $PMGNWPL,6940.894,N,01858.471,E,0000045,M,POI001,,a*3A
SW $PMGNWPL,6940.882,N,01858.446,E,0000045,M,POI001,,a*39
SE $PMGNWPL,6940.871,N,01858.467,E,0000041,M,POI001,,a*32
# NEDRE LYSTHUS
NE $PMGNWPL,6940.874,N,01858.578,E,0000039,M,POI001,,a*37
NW $PMGNWPL,6940.888,N,01858.521,E,0000037,M,POI001,,a*36
SW $PMGNWPL,6940.876,N,01858.492,E,0000044,M,POI001,,a*3A
SE $PMGNWPL,6940.860,N,01858.545,E,0000044,M,POI001,,a*36
# UB
NE $PMGNWPL,6940.868,N,01858.469,E,0000040,M,POI001,,a*35
NW $PMGNWPL,6940.884,N,01858.412,E,0000042,M,POI001,,a*39
SW $PMGNWPL,6940.873,N,01858.364,E,0000064,M,POI001,,a*33
SE $PMGNWPL,6940.855,N,01858.402,E,0000077,M,POI001,,a*32
# ISV
NE $PMGNWPL,6940.840,N,01858.516,E,0000033,M,POI001,,a*32
NW $PMGNWPL,6940.860,N,01858.466,E,0000040,M,POI001,,a*32
SW $PMGNWPL,6940.833,N,01858.358,E,0000082,M,POI001,,a*30
SE $PMGNWPL,6940.803,N,01858.425,E,0000059,M,POI001,,a*38
# TEORIFAG-VEST
NE $PMGNWPL,6940.913,N,01858.317,E,0000079,M,POI001,,a*3C
NW $PMGNWPL,6940.941,N,01858.210,E,0000075,M,POI001,,a*31
SW $PMGNWPL,6940.909,N,01858.129,E,0000057,M,POI001,,a*34
SE $PMGNWPL,6940.862,N,01858.192,E,0000054,M,POI001,,a*3B
# TEORIFAG-ØST
NE $PMGNWPL,6940.883,N,01858.388,E,0000049,M,POI001,,a*31
NW $PMGNWPL,6940.903,N,01858.326,E,0000065,M,POI001,,a*32
SW $PMGNWPL,6940.850,N,01858.213,E,0000052,M,POI001,,a*36
SE $PMGNWPL,6940.845,N,01858.258,E,0000049,M,POI001,,a*37
# RIKSARKIVET
NE $PMGNWPL,6940.861,N,01858.353,E,0000033,M,POI001,,a*36
NW $PMGNWPL,6940.873,N,01858.345,E,0000053,M,POI001,,a*34
SW $PMGNWPL,6940.845,N,01858.290,E,0000059,M,POI001,,a*32
```

```
SE $PMGNWPL,6940.834,N,01858.299,E,0000052,M,POI001,,a*36
# HOVEDGÅRDEN
NE $PMGNWPL,6940.762,N,01858.331,E,0000047,M,POI001,,a*33
NW $PMGNWPL,6940.813,N,01858.230,E,0000054,M,POI001,,a*36
SW $PMGNWPL,6940.796,N,01858.201,E,0000058,M,POI001,,a*3A
SE $PMGNWPL,6940.766,N,01858.296,E,0000051,M,POI001,,a*32
```

A.3 Sporlogg

Nedenfor følger en forklaring på de forskjellige feltene som GPS-en lagrer. Leses fra venstre mot høyre, uttrykket er kommaseparert:

```
$PMGNTRK,6941.578,N,01900.485,E,00018,M,111128.20,A,,031006*74
```

1. Angir protokollen
2. Breddegrad, gitt på grader, minutter og desimalminutter
3. Angir nordlig eller sørlige breddegrad
4. Lengdegrad
5. Angir om det er østlig eller vestlig lengdegrad
6. Elevasjon
7. Angir om elevasjon er uttrykt i meter eller fot
8. Tidspunkt posisjonen ble tatt på, angitt i UTC
9. Angir om datane er validerte
10. Plass for navn på ruta
11. Dato angitt i UTC

```
$PMGNTRK,6941.578,N,01900.485,E,00018,M,111128.20,A,,031006*74
$PMGNTRK,6941.634,N,01900.398,E,00018,M,111228.53,A,,031006*73
$PMGNTRK,6941.654,N,01900.399,E,00033,M,111345.52,A,,031006*76
$PMGNTRK,6941.622,N,01900.404,E,00034,M,111549.52,A,,031006*79
$PMGNTRK,6941.618,N,01900.318,E,00035,M,111623.52,A,,031006*74
$PMGNTRK,6941.591,N,01900.271,E,00030,M,111659.53,A,,031006*71
$PMGNTRK,6941.567,N,01900.221,E,00038,M,111733.53,A,,031006*78
$PMGNTRK,6941.559,N,01900.156,E,00056,M,111901.53,A,,031006*71
$PMGNTRK,6941.506,N,01900.000,E,00061,M,112102.54,A,,031006*72
$PMGNTRK,6941.486,N,01859.909,E,00056,M,112404.54,A,,031006*71
$PMGNTRK,6941.472,N,01859.771,E,00055,M,112540.54,A,,031006*79
$PMGNTRK,6941.487,N,01859.570,E,00064,M,112712.55,A,,031006*76
$PMGNTRK,6941.522,N,01859.428,E,00076,M,112824.55,A,,031006*7D
$PMGNTRK,6941.568,N,01859.322,E,00082,M,112934.57,A,,031006*77
$PMGNTRK,6941.603,N,01859.297,E,00073,M,113148.57,A,,031006*7A
$PMGNTRK,6941.616,N,01859.196,E,00075,M,113316.56,A,,031006*72
$PMGNTRK,6941.648,N,01859.127,E,00080,M,113406.57,A,,031006*7E
$PMGNTRK,6941.646,N,01859.042,E,00078,M,113447.57,A,,031006*70
```

```
$PMGNTRK,6941.670,N,01858.931,E,00090,M,113547.56,A,,031006*7F
$PMGNTRK,6941.621,N,01858.891,E,00092,M,113748.57,A,,031006*7E
$PMGNTRK,6941.569,N,01858.883,E,00069,M,113841.57,A,,031006*70
$PMGNTRK,6941.536,N,01858.800,E,00075,M,113933.57,A,,031006*78
$PMGNTRK,6941.509,N,01858.740,E,00074,M,114017.59,A,,031006*78
$PMGNTRK,6941.477,N,01858.723,E,00076,M,114058.60,A,,031006*76
$PMGNTRK,6941.456,N,01858.742,E,00077,M,114228.59,A,,031006*7C
$PMGNTRK,6941.437,N,01858.696,E,00074,M,114446.59,A,,031006*7E
$PMGNTRK,6941.405,N,01858.641,E,00075,M,114530.12,A,,031006*7B
$PMGNTRK,6941.387,N,01858.553,E,00075,M,114619.61,A,,031006*7A
$PMGNTRK,6941.312,N,01858.373,E,00086,M,114810.59,A,,031006*72
$PMGNTRK,6941.277,N,01858.340,E,00083,M,114908.59,A,,031006*7D
$PMGNTRK,6941.247,N,01858.267,E,00076,M,114951.59,A,,031006*7C
$PMGNTRK,6941.218,N,01858.192,E,00086,M,115104.58,A,,031006*78
$PMGNTRK,6941.193,N,01858.126,E,00088,M,115232.60,A,,031006*74
$PMGNTRK,6941.155,N,01858.070,E,00089,M,115322.60,A,,031006*7D
$PMGNTRK,6941.119,N,01857.971,E,00063,M,115545.61,A,,031006*70
$PMGNTRK,6941.094,N,01857.906,E,00075,M,115628.61,A,,031006*7B
$PMGNTRK,6941.054,N,01857.893,E,00075,M,115717.61,A,,031006*77
$PMGNTRK,6941.034,N,01857.955,E,00073,M,115803.60,A,,031006*77
$PMGNTRK,6941.012,N,01857.952,E,00071,M,115914.59,A,,031006*7B
$PMGNTRK,6940.981,N,01857.998,E,00058,M,120030.60,A,,031006*77
$PMGNTRK,6940.957,N,01858.023,E,00062,M,120110.60,A,,031006*70
$PMGNTRK,6940.960,N,01858.147,E,00064,M,120217.59,A,,031006*7F
$PMGNTRK,6940.977,N,01858.215,E,00065,M,120335.60,A,,031006*77
$PMGNTRK,6940.959,N,01858.288,E,00054,M,120411.61,A,,031006*7D
$PMGNTRK,6940.942,N,01858.370,E,00043,M,120506.60,A,,031006*71
$PMGNTRK,6940.899,N,01858.286,E,00041,M,120620.14,A,,031006*78
$PMGNTRK,6940.887,N,01858.334,E,00056,M,120726.59,A,,031006*77
$PMGNTRK,6940.856,N,01858.318,E,00073,M,120758.60,A,,031006*71
$PMGNTRK,6940.831,N,01858.300,E,00044,M,120901.60,A,,031006*7F
$PMGNTRK,6940.821,N,01858.343,E,00047,M,121030.60,A,,031006*70
$PMGNTRK,6940.832,N,01858.379,E,00036,M,121211.61,A,,031006*7D
$PMGNTRK,6940.827,N,01858.498,E,00047,M,121249.61,A,,031006*7A
$PMGNTRK,6940.852,N,01858.515,E,00043,M,121412.60,A,,031006*71
$PMGNTRK,6940.903,N,01858.566,E,00035,M,121617.56,A,,031006*73
$PMGNCMD,END*3D
```