Norges miljø- og biovitenskapelige universitet

Masteroppgave 2016 30 stp

Norges miljø- og biovitenskapelige universitet
Fakultet for miljøvitenskap og teknologi
Institutt for matematiske realfag og teknologi

Gruppedetektering på GNSS-spor ved bruk av minimum spenntre

Group Detection on GNSS Based Tracks Using Minimum Spanning Tree

Mohammad Usman
Master i geomatikk

Forord

Jeg vil rette en stor takk til min veileder Håvard Tveite, for den verdifulle hjelpen i arbeidet med denne oppgaven. Oppgaven ble fullført takket være hans genuine interesse for å hjelpe, tålmodighet og kloke råd. Jeg vil også takke alle venner og medstudenter for det gode selskapet og støtte i skrivingen av oppgaven. Særlig Umair Iqbal og Samir Adrik som har hjulpet til med gjennomlesing i innspurten.

Til slutt vil jeg takke mor, far og resten av familien for alltid å være positive og støtte meg til tross for alle utsettelsene.

Ås, 18. mai 2016

Sammendrag

I denne oppgaven er det anvendt gruppedetektering på GNSS-spor fra et orienteringsløp. Ved å betrakte bevegelseshistorikk som punktbevegelse, er problemet redusert til å finne gjentagende romlig naboskap mellom to eller flere punkter som kan antas å være i gruppe. Det ble implementert et program som utfører iterativ clustering gitt ved minimum spenntre, og oppdaterer datasettet med felles attributt for hvert *cluster*. Basert på dette attributtet ble det utført manuell filtrering i SQL for å bestemme *clustere* som varte lenge nok til å kunne regnes som grupper. Det ble også laget en visualisering av strekninger hvor de antatte gruppene var i bevegelse. Resultatene viser at en kan identifisere mulige grupper av individer, selv på et datasett hvor det totalt sett var tett bevegelse. Det ble også funnet at metoden er svak på å kartlegge vekslinger og brudd i gruppene.

Abstract

In this thesis, an application of group detection has been tested on GNSS-tracks from orienteering. By considering historic movement as moving points, the problem is generalized to finding two or more neighbouring points who maintain their neighbourhood repeatedly over time. A program was developed for iteratively determining clusters using minimum spanning tree and updating the data set with a common attribute for cluster members. This was followed by a filtering step in SQL to find the clusters with duration long enough to be considered as groups. For groups in movement a visualization were also made. The results show that it is possible to identify possible groups of people, even when the data is about overall dense movement. The method was found to be weak in identifying splits and transitions between groups.

Innhold

Figurer

Tabeller

1. Introduksjon

Som en av anvendelsene til mobil posisjonsbestemmelse (GPS mm.), er kartlegging av bevegelse blitt meget aktuelt de siste årene. Blant eksemplene er overvåkning og navigering av selvgående og fjernstyrte fartøy, i tillegg til sporing av bevegelser hos mennesker og dyr. Bevegelse i rom og tid kan oversettes til atferd med tanke på hvor tidsbruken skjer [6]. Når dette er tallfestet med stor nøyaktighet i form av bevegelsesspor, kan det avsløres situasjoner som ellers er vanskelig eller umulig å fange opp. Det er særlig interessant å følge bevegelsene til mennesker og dyr fordi de kan ha både rutinemessig og (i ulik grad) vilkårlig bevegelse mellom steder.

I mange daglige situasjoner skjer det raske endringer i hvor folk befinner seg, med kontinuerlig forflytning og posisjonering i forhold til andre. Både individuelle og kollektive bevegelsesmønstre er gode utgangspunkt for å identifisere trender og statistikk over stedsbruk. Disse kan gi svar på hvor ett individ var på gitte tidspunkt, men også om hvem som var i nærheten av hverandre og når dette var tilfelle. At grupper av folk dannes på visse steder og tidspunkter kan være svært nyttig informasjon, både for kommersiell virksomhet, forskning og sikkerhetstiltak. Påvirkende faktorer kan være for eksempel steder og ruter av felles interesse, eller at personer naturlig tilhører en gruppe (som partnere, kolleger, venner osv.). For eksempel kan turismenæringen dra nytte av dette ved å se på hvilke steder som ofte besøkes og hvordan trafikken av folk skjer fra sted til sted [14]. Dette kan bidra i planlegging av nye attraksjoner og utbedringer av de eksisterende besøkspunkt for mer effektiv inn- og utfart. Både i Forsvaret og ulike idrett kan det brukes tilsvarende teknikk for å forbedre strategier på oppgaver som krever samarbeid og de som må løses individuelt.

Bruk av kun posisjonshistorikk til å kartlegge atferd, er ikke en triviell oppgave. Mye forskning er gjort på dette området, da man har forstått tidlig at rom-tid informasjon av denne typen kan brukes til mye mer enn det som kan tolkes direkte. Laube et al. [12] er blant de som utviklet kvantitative metoder for å identifisere spesifikke atferdstyper, som bevegelse i flokk og ledere og følgere. En oversikt over eksisterende litteratur på dette området finnes i [13] hvor det også kommer fram at etablerte GIS mangler støtte for tidsvarierende data. Nye teknikker for visualisering av naboskap mellom flere dekkes av [2]. På grunn av de kompliserende faktorene i håndtering av tidsvarierende informasjon, kreves det ofte tilpassede metoder for å finne de spesifikke bevegelsesmønstre. Et eksempel er implementeringen i Giannotti et al. [8]. Deres hovedpoeng er å trekke ut atferdsrelaterte bevegelsesmønstre ut fra store og komplekse datasett. I tillegg til at metoder utvikles for bestemte formål innen bevegelsesanalyse, er i følge Long and Nelson [13] teknikkene for å bestemme bevegelsesmønstre som angår grupper og gruppedynamikk lite testet på virkelige datasett.

1.1. Problemstilling

Oppgaven tar for seg identifisering av grupper ut fra bevegelsesspor, uten eksplisitt tilleggsinformasjon. Det blir forsøkt å svare på følgende:

Gitt at en samling av mennesker har gått fra felles start til felles mål; Hvor godt kan en identifisere mulige grupper blant dem når deres bevegelse betraktes som punktbevegelse?

2. Teori

2.1. Posisjonsdata for analyse av bevegelse

Rom-tid data som er ment for å representere bevegelse kalles ofte *movement data*, og referer til bevegelse på eller nær jordoverflaten [13]. Det som er kjent som bevegelsesspor («GPS-spor») inngår i denne kategorien. Hvis et bevegelsesspor skulle gjenspeile bevegelse fullstendig lik virkeligheten, måtte det ha vært en kontinuerlig *Trajectory* mellom et start- og sluttidspunkt[3]. I dette ideelle tilfellet ville man kunne besvare spørsmål som «hvor var personen ved tidspunkt t» for alle mulige tidspunkt så lenge de var innenfor start- og sluttidspunkt. Som det generelle tilfellet med geografisk data, er ideen også her å fremstille en modell av virkelige fenomener i verden.

Realisering av *Trajectory* må skje på diskret form, det vil si en serie av punktkoordinater med tilhørende tidsstempel på formen $< p_i, t_i >$ [3, 16]. Disse øyeblikksbildene av posisjoner følger vektorrepresentasjon kjent innen GIS, hvor minste enhet er punktobjekt. Det er vanlig å visualisere bevegelsesspor som polylinjer for å beskrive en «rute» et objekt eller et individ kan ha gått. For eksempel kan bevegelsene til en skiløper fremstilles på denne måten. Videre i oppgaven blir begrepene bevegelsesspor og GPS-spor brukt om hverandre når det henvises til posisjonshistorikk på Trajectory-formen. Movement data blir kalt posisjonsdata i resten av dokumentet. Begrepet er lånt fra [14], hvor det kalles *positioning data.*

En viktig faktor som gjelder all posisjonsdata er samplingsrate, det vil si hvor ofte etterfølgende posisjoner er målt. Begrepet *scale* (tidsoppløsning) blir også brukt om dette[13]. Andrienko et al. [3] kategoriserer flere *samplingsregler* som kan benyttes under datafangst, deriblant tidsstyrt (time based) og endringsstyrt (change based) sampling. Tidsstyrt sampling betyr at det er fast tidsintervall (f.eks hvert 3. sekund) mellom observasjonene som registreres. Endringsstyrt sampling derimot, innebærer at posisjon og tidspunkt legges til kun hvis den tilsier bevegelse i forhold til den forrige registrerte. Sistnevnte kan gi irregulær samplingsrate, samtidig som det ikke registreres unødvendig mange observasjoner.

Ved bearbeiding av posisjonsdata, er spørringer på tid og tidsperioder kanskje det viktigste man gjør ved problemstillinger som gjelder bevegelsesmønster. På grunn av diskret registrering av posisjoner, vil det alltid mangle informasjon mellom kjentpunktene. Dersom vi er ute etter posisjon ved et vilkårlig tidspunkt t innenfor den første målingen t_{start} og den siste måling t_{slutt}, kan det derfor være nødvendig å estimere koordinater ut fra de nærmeste registrerte. Til dette kan man bruke lineær interpolering, som er gitt ved [16]:

$$x, y = x_1 + (t - t_1)\left(\frac{x_2 - x_1}{t_2 - t_1}\right), y_1 + (t - t_1)\left(\frac{y_2 - y_1}{t_2 - t_1}\right) \tag{2.1}$$

hvor (x, y) er interpolerte koordinater ved tidspunkt t. De nærmeste kjentpunktene observert før og etter t er (x_1, y_1) og (x_2, y_2) ved tidspunkt t_1 og t_2 henholdsvis. Lineær interpolering antar rettlinjet og konstant bevegelse mellom kjentpunkter [16]. Dette medfører at den kun er gyldig når tidsforskjellen mellom etterfølgende observasjoner er liten i forhold til hvor mye bevegelsene kan endre seg [1]. For eksempel gir det liten mening å legge inn kunstige punktkoordinater på denne måten når det er et gap på flere minutter i posisjonsdata fra en løpeidrett. Med interpolering kan man blant annet utføre kvantitative

analyser på ønsket tidsoppløsning. Figur 2.1 viser et eksempel på et bevegelsesspor som er resamplet [1] ved hjelp av lineær interpolering.

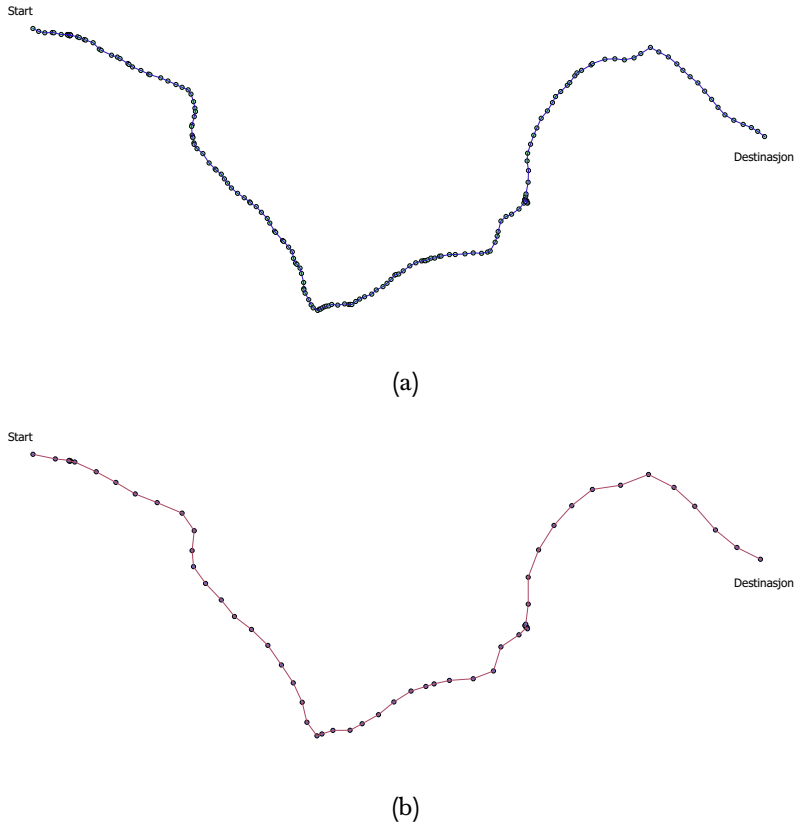


Fig. 2.1.: (a) Opprinnelig GPS-spor med 185 punktobservasjoner og irregulær samplingsrate (b) resamplet variant hvor antall punkter er redusert til 57 ved en regulær samplingsrate.

2.1.1. Referansesystem

Hvorvidt man skal benytte geografiske koordinater (lengdegrad,breddegrad) eller projiserte koordinater (x,y i meter), avhenger av bruksområde og analyseteknikk. Geografiske koordinatsystemer er velkjent og kan brukes til å angi posisjon entydig og med stor nøyaktighet hvor som helst i verden. Prinsippet bak geografiske koordinatsystem kan forklares ved å anta at Jorda er kuleformet, og origo er i sentrum av Jorda. Nullpunkt for breddegrader er ekvator, dvs at breddegrader er den vertikale vinkelen mellom en imaginær linje fra origo relativt ekvatorplanet. Lengdegrader er horisontal vinkel mellom en linje fra origo relativt en sentralmeridian. Fordi Jorda ikke er fullstendig kuleformet (ellipsoide), er referansesystem basert på *datum*, matematiske modeller for å beskrive Jordas form og plassering av de tre aksene i et kartesisk koordinatsystem og en rekke andre paremetere til geodetiske formål. Et eksempel på et datum er WGS84 [17, kap.2.1], som bruker en referanseellipsoide med origo i Jordas massesenter og førsteaksen går gjennom sentralmeridianen. Sentralmeridianen i et globalt referansesystem som WGS84 er Greenwich-meridianen som er definert som 0 breddegrader.

Projiserte koordinater eller plankoordinater baserer seg på en kartprojeksjon, avbildning av ellipsoiden på et todimensjonalt plan. En av de kjente kartprojeksjonene, UTM er basert på *transversal mercator* [22]. Denne projeksjonen baserer på at det todimensjonale planet er imaginær liggende sylinder rundt ellipsoiden, som tangerer ellipsoiden langs en sentralmeridian (nord-sør) og matematiske modeller brukes for å projisere punkter over på denne sylinderen. UTM er delt inn soner og hver sone dekker 6 lengdegrader(med noen får unntak). Hver sone i UTM er en ny projeksjon med egen sentralmeridian for å minske fortegningen

øst-vest. I Norge brukes enheten meter for å oppgi plankoordinater, men andre enheter som fot, yard, km osv kan også brukes.

Geografiske koordinater kan brukes direkte dersom formålet kun er visualisering, og når strekningene i bevegelsessporene er svært lange, slik at et lokalt koordinatsystem ikke er hensiktsmessig å bruke. Avstandsberegninger er derimot enklere å gjøre på projiserte koordinater, da en generell avstandsfunksjon kan brukes direkte på koordinatene.

2.1.2. Datainnsamling

Sporing av bevegelse kan gjøres ved hjelp av satellittbasert posisjoneringsteknologi, kjent som Global Navigation Satellite System (GNSS). Mange GNSS-mottakere kan automatisk logge og lagre posisjoner over tid. Koordinatene mottas typisk på formen breddegrad, lengdegrad og høyde over ellipsoiden (latitude, longitude, altitude) i WGS84-systemet.

Satellittbasert posisjonsbestemmelse kan regnes som den foretrukne metoden for sporing av flere grunner. GNSS er designet for global dekning av satellittsignaler, slik at det til enhver tid kan oppnås kontakt med minst 4 navigasjonssatellitter hvor som helst på Jorda(Avstandsmålinger til satellitt og gangtid for signalet er med på å bestemme tredimensjonale koordinater til en mottaker, noe som krever 4 satellitter). For brukere av moderne smarttelefoner, kreves det svært lite innsats og kostnad å bruke GNSS-posisjonering. Flere apper finnes for «GPS-tracking» og lagring både i Android og iOS-miljø. GNSS-målinger kan gi koordinatverdier med svært god nøyaktighet under gode forhold.

Et alternativ til GNSS, som er like tilgjengelig og også kan brukes med mobiltelefoner er GSM-basert posisjonering. Fordelen med denne teknikken er at den også fungerer innendørs. Den gir derimot betydelig mindre nøyaktige resultater enn GNSS, som det fremgår i [18]. GSM kan i stedet benyttes som et supplement til GNSS for å motta GPS-korreksjoner i sanntid for å oppnå høyere nøyaktighet (se Assisted GPS i [5]) og dersom en foretar sanntidssporing, sende GPS-data til en sentral enhet over mobilnettet.

2.1.3. Lagringsformat

Posisjonslogg fra mobile enheter kan lagres lokalt for senere bearbeiding og analyse. GNSS-enheter som støtter sporing, har også rutiner for lagring av data i utvekslingsvennlig format. Et av disse er GPS Exchange Format (GPX)[19]. Innholdet i en GPX-fil er basert på XML-standard, dvs at informasjonen er oppdelt ved HTML-lignende *tags* på ulike nivåer. Informasjonen lagres på hierarkisk sett. Det benyttes såkalte *parent* og *child* elementer, som beskriver forhold mellom objekter(en ting tilhører en kategori, et element består av flere mindre enheter). GNSS-spor lagres som et *track* som består av ett eller flere *tracksegments* dvs, sammenhengende spor; som igjen består av koordinater og tidsstempel som en rekke *trackpoint* ordnet etter tidsstempel. Ved tap av signal til satellittene, avsluttes et *tracksegment*, og et nytt startes når signalet eventuelt er tilbake. Et minimalt eksempel er skissert i figur 2.2.

Alternativet til GPX-format er KML, som også følger samme prinsipp men er tilpasset Google Maps/Google Earth miljø. Slik filstruktur er svært godt egnet for utveksling av informasjon, og har god lesbarhet både manuelt og maskinelt.

```
<?xml version=1.0' ...>
  <gpx version=1.1 ...>
    ... (metadata)
    <trk>
        <name>25. jul. 2015 20.46.50</name>
            ...
            <trkseg>
                <trkpt lat="59.939968" lon="10.760075">
                    <ele>116</ele>
                    <time>2015-07-25T18:46:51.000Z</time>
                </trkpt>
                <trkpt lat="59.940048" lon="10.760167">
                    <ele>128</ele>
                    <time>2015-07-25T18:47:06.000Z</time>
                </trkpt>
            </trkseg>
    </trk>
  </gpx>
```

Fig. 2.2.: Eksempel på GPX-struktur

2.1.4. Feilkilder og kontekst

Usikkerheter knyttet til målemetode bør tas med i betraktning ved arbeid med posisjonsdata. I tilfellet med GNSS-målte posisjoner er feilbudsjettet bestående av følgende [22]:

Usikkerhet i satellittenes klokker. Selv med de nøyaktige atomklokkene om bord på navigasjonssatellittene, kan en liten feil bidra med stort avvik i koordinatmålingene. Feil på nanosekunder kan føre til en feil i størrelsesorden meter.

Banefeil. Posisjonen til satellittene er kjent med nøyaktighet på meternivå.

Målt gangtid av signalet. Unøyaktighet i mottakerens klokke fører til feil i beregning av signalets gangtid og dermed avstand til satellitt.

Multipath. Signalet kan treffe et annet objekt før det ankommer mottakeren. Dette vil gi feil avstandsmåling og dermed feil i bestemmelse av posisjonen til mottakeren.

I tillegg kommer Dilution of Precision (DOP), som følge av konstellasjon til sattellittene som kommuniserer med mottakeren. Det ideelle er stor spredning (lav DOP). Ved observasjoner i bevegelse vil datasettet ofte bestå av mange koordinater. Med nærmest kontinuerlige målinger er det derfor stor sannsynliget for avvik å målingene (f.eks multipath-effekt i skogområder).

I følge Andrienko et al. [1] er det viktig å ta hensyn til kontekst, dvs bakgrunnsinformasjon om bevegelsen som finner sted. Den kan si noe om egenskapene til objektene som studeres, og deres omgivelser. Den kan også inkludere mulige hendelser relatert til både sted, og objekter (f.eks mennesker). Når en gjør avstandsberegninger for å identifisere en mulig hendelse, for eksempel møte mellom personer, er det derfor viktig å både ta hensyn til usikkerhet i dataene og hvor rimelig antagelsen er i forhold til antatt visuell avstand mellom mennesker. Terrengtype spiller også inn i et slikt tilfelle. I denne oppgaven arbeides ut fra posisjonsdata uten bakgrunnsinformasjon om feil i målingene. I det følgende antas det at målingene er uten feil. Dette fordi fokuset er på prinsippene i gruppedetektering.

2.2. Gruppedetektering

Gruppedetektering i posisjonsdata er en oppgave som kan generaliseres til søk etter *moving clusters* [6]. Prinsippet i alle tilfeller er å klassifisere individer eller objekter som reiser sammen i en gruppe. En gruppe kan defineres på ulike måter basert på regler som skal bestemme *clustere* («klynger»). Når *clustere* er identifisert, må de følges over tid for å kontrollere varighet og holde oversikt over medlemmene.

Et av de tidlige konseptene for gruppedetektering i litteraturen er kjent som *flokk* utviklet av Laube et al. [12] og senere utvidet av Benkert et al. [4] av hensyn til blant annet varigheten til gruppebevegelsen i en flokk. En flokk er definert ved tre parametere: antall medlemmer m, antall tidsenheter k i intervallet $[t_i, t_j]$ og en radius r i et sirkulært område [4]. Posisjonene til alle m individene må befinne seg innenfor en sirkel med radius r over alle de k tidsenhetene. Figur 2.3 viser et eksempel på en flokk bestående av tre medlemmer. Merk at søk gjøres fra og med t_1 til og med t_4 i diskrete tidssteg(Se samplingsrate og interpolering i 2.1).

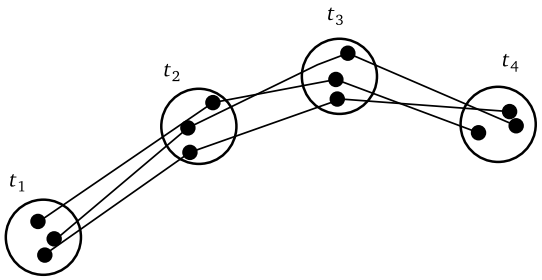


Fig. 2.3.: Gruppe av punktobjekter som kan regnes som flokk[4].

Denne tilnærmingen har fått kritikk for mangel på fleksibilitet med tanke på utstrekningen til posisjonene. I virkeligheten kan ikke en gruppe nødvendigvis bestemmes med sirkulær avgrensning. Et typisk eksempel på situasjoner som kan gi unøyaktige og feil resultater er bevegelse i en rekke. Valg av for liten radius, kan utelukke medlemmer og kun gi deler av en gruppe. I motsatt fall kan man risikere å inkludere feilaktig individer som ikke inngår i en gruppe. Alle disse argumentene brukes av Jeung et al. [10] som motivasjon i det de presenterer det mer fleksible konseptet for gruppe, kalt konvoi.

På overordnet nivå, kan en konvoi beskrives som en gruppe med objekter som har en sekvensiell romlig kobling med hverandre over en tidsperiode [10]. Dette skal sikre et man oppdager grupper med vilkårlig utstrekning eller konstellasjon. Bestemmelse av en konvoi i et sett med bevegelsesspor involverer parameterne m, e og k. Minst m objekter må over en tidsperiode på minimum k diskrete enheter være *density-connected*(koblet via hverandres nabolag og nabotetthet, se 2.2.1) med hensyn på avstandsterskel e. Jeung et al. [10] dekker dermed følgende viktige problemstillinger som kan gjelde for virkelige grupper i rom-tid:

Vilkårlig og variabel utstrekning. Deres valg av *cluster*-funksjon (DBSCAN, forklares senere) dekker tilfeller av *clustere* hvor punkter ikke nødvendigvis må ligge innenfor en sirkulær buffer, men kan også ha mange andre former (inkludert en lang rekke, L-form osv.). Man tar med dette også hensyn til at utstrekning kan endre seg over tid, og at det likevel er en «link» mellom eventuelle gruppemedlemmer.

Varighet. I sjekktidspunkter sammenlignes etterfølgende varianter av hvert *cluster* for å kontrollere samsvar av medlemskap fra et tidspunkt til det neste og de etterfølgende. I tillegg til at gruppen må være på en viss størrelse, må medlemmene bli «sett» sammen gjentatte ganger.

Faste og variable gruppestørrelser. Tilfellene med fast gruppesammensetning og der kan komme nye medlemmer til en eksisterende gruppe, samt at noen kan forlate en gruppe etter en tid er også dekket i konvoi-konseptet.

I de følgende to avsnittene gis det en kort beskrivelse av *Density Based Clustering* og *clustering* basert på minimum spenntre. Begge disse metodene kan finne *clustere* med vilkårlig utstrekning, men har ulike kriterier for å bestemme sekvenser av punkter som inngår i *clustere*.

2.2.1. Density Based Clustering

Ideen om *density connection* stammer fra *clustering*-metoden DBSCAN [7] som baserer seg på at et sirkulært nabolag fra hvert punkt kan gi en sekvens av nabopunkter slik at de enten kan klassifiseres som *clustere* eller støy. Nabolag bestemmes ved gitt radius e og punkttetthet gir naboforbindelser. Når to punkter p og q er *density-connected* må følgende være oppfylt [7]: Det finnes en sekvens med N punkter $o_1 ... o_N$ mellom p og q som alle har nabolag $>= n$, og hvor nabolaget til o_i inneholder o_{i-1} og o_{i+1} samtidig som nabolaget til o_1 inneholder p og nabolaget til o_N inneholder q.

Dette betyr at p og q kan være endepunkter (*border points*) ved at de selv ikke må ha $>= n$ naboer, men inngår som en av naboene til et annet punkt o (*core point*) som oppfyller kravet om nabostørrelse. p og q kan selv også være *core points* når de har nabostørrelse $>= n$ i likhet med alle punktene i deres sirkulære nabolag. Punkter som ikke er *density connected* blir regnet som støy (*noise points*) da de ikke nås fra andre punkter med de nevnte kravene. Figur 2.4 viser to *clustere* funnet ved DBSCAN. Merk at de gule *clusteret* består utelukkende av *core points*, mens det røde har 3 endepunkter.

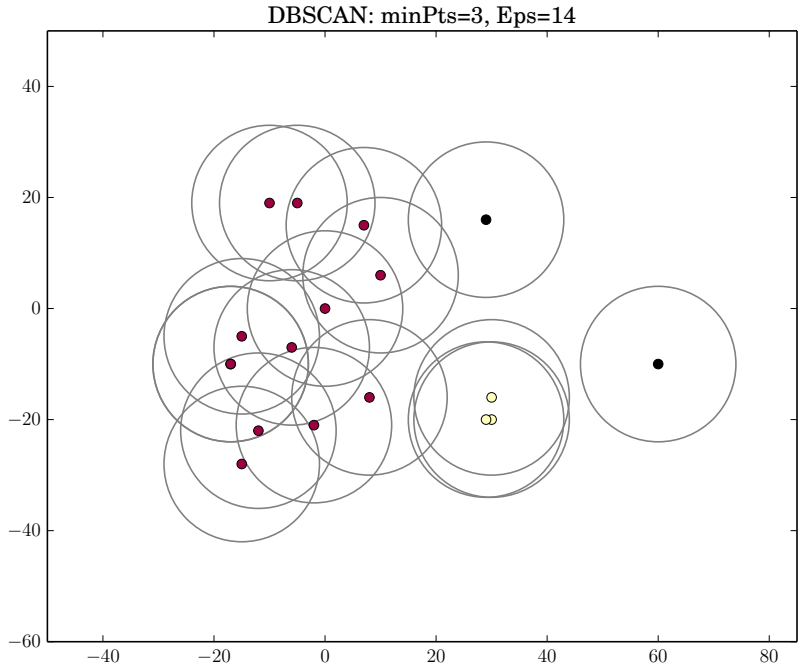


Fig. 2.4.: Eksempel på kjøring av DBSCAN. To *clustere* er produsert. To av punktene (svarte) tilhører ingen av clustrene og blir regnet som støy

En konvoi baserer seg på DBSCAN [7], som står for Density Based Spatial Clustering of Applications with Noise. En konvoi er et slikt *cluster* med et visst antall objekter. Ett enkelt *cluster* utgjør en gruppe når det består gjennom flere tidsenheter som allerede nevnt.

2.2.2. Clustering basert på minimum spenntre

I en sammenhengende og urettet graf $G = (n, k)$ er et spenntre gitt ved alle nodene i G forbundet via kanter slik at det ikke forekommer løkker eller sykler[kap 23.4][11]. Det vil si, fra en hvilken som helst node er det en "vei" til en vilkårlig annen node på en slik form at hver node kun besøkes en gang. Fordi det ikke er noen syklus i et spenntre, består det av $n-1$ kanter. En graf kan inneholde flere enn ett spenntre. I den vektede varianten av G er minste spenntreet det spenntreet som har minste totale kostnad (vekt) og kalles minimum spenntre tree [9].

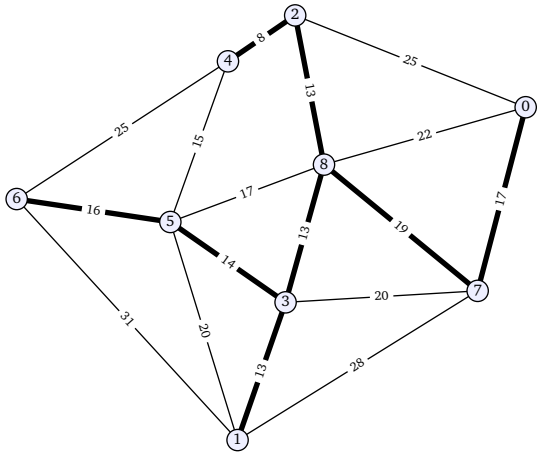


Fig. 2.5.: Et minimum spenntre. Den uthevede subgrafen er spenntreet som omfatter alle nodene i grafen med minste totale kostnad. I dette tilfellet er kantlengdene (og kostnad) proporsjonale med avstanden, slik at det blir et euklidsk minimum spenntre.

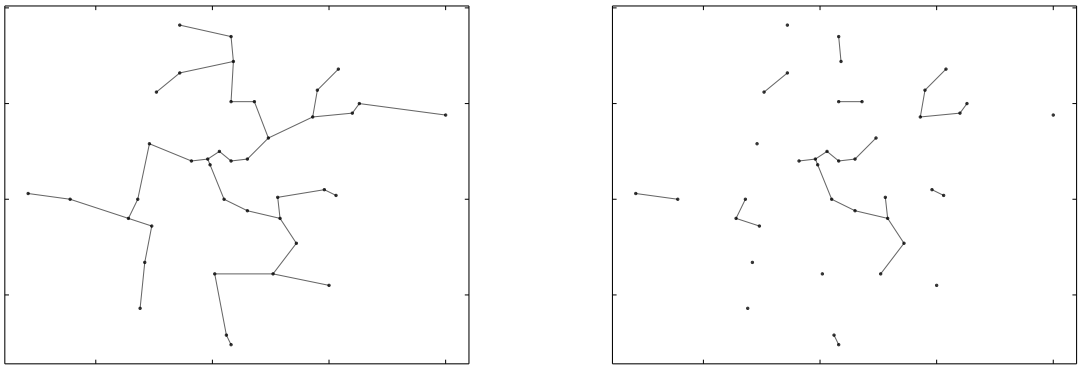
Euklidsk minimum spenntre er spesialtilfelle hvor kantlengdene representerer euklidske avstander mellom nodene som forestiller posisjoner i metrisk rom [9]. Et eksempel er vist i figur 2.5. Bestemmelse av nabopar ved minimum spenntre konstruksjon sikrer et optimalt nettverk med hensyn på minimums-kravet. Dette gjør at minimum spenntre har mange bruksområder. Planlegging av veinettverk og el-nett er vanlige eksempler.

Det har lenge vært kjent at kostminimeringsegenskapene til minimum spenntre også kan brukes for *cluster*-analyse [21]. Med geografiske data kan dette forstås intuitivt. I første omgang kan en tenke på minimum spenntreet i figur 2.5 som den korteste mulige kjeden som forbinder alle punktene. Videre kan man tenke seg at jo lengre avstand mellom nabopar, jo svakere er leddet i kjeden. Fjerning av de uønskede leddene resulterer i flere subtrær som er adskilt fra hverandre.

Med et minimum spenntre som utgangspunkt er det to grunnleggende tilnærminger for å produsere *clustere* [20]: Fjern alle kanter hvis vekt overskrider en gitt maksgrense, eller fjern så mange av lengste kantene nødvendig for å produsere et gitt antall *clustere*. Eksempelet i figur 2.6 illustrerer tilfellet med eliminering av kanter som har vekt større enn et maksimumskriterium.

I likhet med DBSCAN, kan *clustere* funnet med minimum spenntre ha vilkårlig form. Tilnærmingene som er presentert over, krever at et minimum spenntre allerede er konstruert før eliminering av «for lange» kanter skjer. Gitt en komplett graf kan minimum spenntre bestemmes ved for eksempel Prims eller Kruskals algoritme [20]:

Prims algoritme starter i en vilkårlig node og former et tre ut fra den. En ny kant legges til dersom den



(a) (b)

Fig. 2.6.: Grunnleggende *clustering* med minimum spenntre med utgangspunkt (a) Minimum spenntre i et punktsett og (b) *clustere* fremstilt ved fjerning av kanter lenger enn en maksverdi.

har minimum vekt i forhold til andre kanter ut fra den aktuelle noden. Dette gjentas til alle noder er lagt til, og kravet om minimum spenntre er oppfylt. I hele prosessen går en sammenhengende tre ut fra rotnoden.

I Kruskals algoritme kobles flere trær sammen til en sammenhengende som tilslutt blir til ett minimum spenntre. I starten er alle nodene separate trær. Den korteste kanten velges ut slik at den nå kobler to trær sammen. En ny kant velges slik at den har minimum vekt blant de gjenværende, samtidig som den er sammenhengende med den eksisterende; og at det ikke forekommer sykler. Dette gjentas til det eksisterer kun ett sammenhengende tre.

3. Metode

3.1. Klargjøring og preprosessering av data

Datasettet i shapefile-formatet ble importert til databasesystemet PostgreSQL før det ble utført koordinattransformasjon fra lengde/bredde-koordinater til x, y i UTM-systemet (sone 32). PostGIS-verktøyet «shapefile and dbf loader» ble brukt til importen og opprettelse av tabell i databasen. Koordinattransformasjonen ble gjort med PostGIS' innebygde funksjon `ST_Transform` og dataene ble samtidig lagt over i en ny tabell med suffiks 'utm32'. Selv om det finnes implementeringer for å beregne avstander og naboskap på geografiske koordinater, var det mest praktisk med todimensjonale (x, y)-koordinater for å generalisere problemet til å gjelde i et hvilket som helst kartesisk 2D-system. UTM sone 32 ble valgt som koordinatreferansesystem fordi det er best tilpasset Sør-norge.

Videre ble det utført interpolering på UTM-datasettet for omgjøring til regulært tidsintervall mellom koordinatobservasjonene i alle bevegelsessporene. Det ble opprettet en ny tabell som skulle inneholde det regulariserte datasettet. En iterativ prosedyre ble utformet i programmeringsspråket Python. Den tok seg av tidsspørringer i faste tidssteg `t_delta` sekunder med første observerte tidspunkt t_s som startverdi. For hvert gjeldende tidspunkt t_i ble det søkt etter observasjoner på eller rundt t_i i hvert enkelt bevegelsesspor. En SQL-spørring hentet ut to sett med observasjoner tilhørende identifikator (navn); ett med nærmeste tidsstempel før og ett med nærmeste etter t_i. I samme iterasjon ble det kjørt en interpoleringsfunksjon ut fra de to settene med koordinater og tidspunkt samt søketidspunkt (direkte bruk av formel 2.1). En kontrollrutine i iterasjonen ignorerte tilfeller hvor det var for stort tidsgap mellom to etterfølgende punkter ut fra gitt parameter `max_gap`. Interpolerte koordinater, tidspunkter t_i samt identifikator ble sendt fortløpende til den nye tabellen i databasen til iterasjonen var ferdig.

En resampling av denne typen sikrer at tidsspørringer kan gjøres jevnt over hele datasettet så lenge tidsintervall er kjent. Interpolering er også nødvendig for sammenligning av posisjoner mellom flere individer på bestemte tidspunkt (som ikke alltid samsvarer med måletidspunkt i data). Mange av observasjonene i det nye datasettet vil derfor bestå av kunstige observasjoner mellom de faktisk observerte. Resampling med interpolering er gjort i et separat steg for å forenkle videre behandling. Det resamplede datasettet kan brukes som input til gruppedetektering (om nødvendig, flere ganger) med hensyn på det nye tidsintervallet.

3.2. Gruppedetektering

Klassifisering av grupper er gjort i flere steg. Ideen var å utnytte egenskapene til databasesystemet PostgreSQL med tanke på aggregering og visualisering. *Clustering* ble gjort ved hjelp av minimum spenntre i en iterativ prosedyre(en oversikt er vist i figur 3.1). I stedet for å sjekke varighet til hvert *cluster*, ble det kun generert en felles identifikator på hver gruppesammensetning ved hvert sjekktidspunkt.

Meningen bak dette er å gi fleksibilitet, da varighet til gruppene sjekkes direkte i SQL, uten å måtte kjøre *clustering* algoritmen flere ganger. Detaljert beskrivelse av den iterative *clusteringen* samt etterbehandlingen i SQL gis i de følgende avsnittene.

3.2.1. Algoritmen MST cluster timeseries

Algoritmen `mst_cluster_timeseries` omfatter operasjonene vist i figur 3.1. Den er implementert i form av et Python-program. Algoritmen tar parameterne `t_delta`, `max_dist`, og `min_size`, som er henholdsvis fast tidsintervall (eks. hvert 10.sekund) for behandling av posisjoner, avstandsgrense for minimum spenntre splitt, og minste antall individer for at det skal regnes som gruppe.

Det forutsettes at datasettet som brukes som input, er resamplet slik at det er faste og samsvarende tidsintervall mellom observasjonene i alle bevegelsesspor. Videre må det inneholde kolonner for navn, x, y og tidsstempel. x og y er øst- og nordkoordinater i meter, og tidsstempel er dato og klokkeslett. Fast tidsintervall i observasjonene kan oppnås ved f.eks ved preprosessering forklart i avsnitt 3.1.

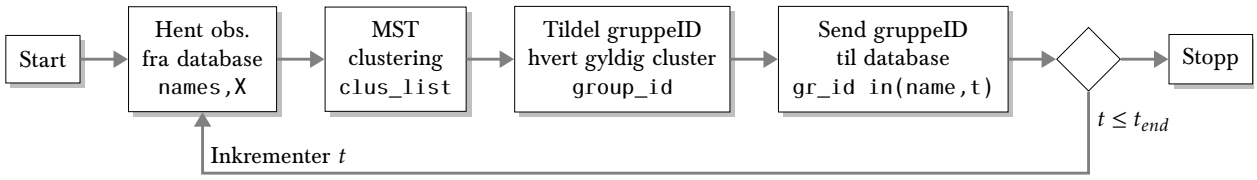


Fig. 3.1.: Sekvensiell klassifisering av grupper ved *clustering* på tidsøyeblikk. Medlemmene som inngår i *clustere* får tildelt felles gruppeidentifikator.

Før iterasjonen begynner, blir det lagt til en ny kolonne i datasettet for lagring av gruppeidentifikator. I hver iterasjon blir det gjort spørringer mot databasen på hvert tidspunkt fra starttidspunkt, til siste registrering, for å hente ut koordinatobservasjoner, behandle dem og lagre gruppeidentifikator i databasen. Det samme datasettet som gis som input, blir oppdatert med gruppeidentifikator etter kjøring av algoritmen.

Verktøy

Den Python- baserte implementeringen er avhengig av flere tredjepartsbibliotek (moduler) for kontakt med database, håndtering av grafstruktur og *clustering*. I tillegg ble det utviklet to egne moduler for ofte brukte funksjoner for kjøring av tilpassede database-operasjoner og konvertering av tidsstempel til og fra enheten sekunder og datostreng. Disse er listet opp i tabell 3.1 sammen med en kortfattet forklaring på hva de brukes til.

Tabell 3.1.: Python moduler brukt i implementeringen

Ressurs	Beskrivelse
psycopg2[1]	Databasehåndtering mot PostgreSQL db-system
sklearn.neighbors.DistanceMetric[2]	Del av scikit-learn for avansert dataanalyse
scipy	Matematisk og vitenskapelig samling av verktøy
networkX[3]	Nettverksanalyseverktøy med en rekke graf-algoritmer
handle_db	Egne funksjoner for databaseoperasjoner
time_conv	Egne funksjoner for konvertering av dato/tidsformat

Implemenringen er gjort i Python version 2.7.6. Tilleggs-bibliotekene måtte installeres for å inkorporere dem i Python kjøremiljø i Windows 8.1 operativsystem. Ressursene som er listet i tabell 3.1 kunne importeres med standard import-setning i python med samme navn som gitt her. Det ble forsøkt å bruke færrest mulig tilleggsressurser av hensyn til fleksibilitet, men samtidig utnytte eksisterende implementeringer for «samlebåndsoppgaver» som for eksempel avstandsmålinger og bestemmelse av naboer i graf.

[1] http://initd.org/psycopg/
[2] http://scikit-learn.org/stable/index.html
[3] https://networkx.github.io/

Initialisering

Clustering-prosedyren starter med å bestemme tidsgrensene for hele datasettet. Etter at det blir opprettet forbindelse med databasen, blir tidsgrensene `tmin` og `tmax` hentet fra databasen. De omgjøres samtidig til antall sekunder siden (epoke) 01.01.1970 00:00:00. Denne omformingen skjer direkte i en SQL-spørring.

Det blir opprettet en tom oppslagstabell for gruppeidentifikator i form av et *dictionary*. Den har levetid så lenge programmet kjører og er ment for systematisk tildeling av gruppeID. Variabelen for gruppeID lagres som heltall (integer) og gis «tom» intitialverdi. Deretter gjøres det en endring i tabellen fra datasettet, ved at det legges til en kolonne for gruppeID. Dette gjøres ved en enkel SQL-setning. Med gitte parametere, `max_dist`, `min_size` og `t_delta` settes iterasjonen i gang fra og med tidspunkt `tmin` med fast økning på `t_delta`.

Tidsspørringer på koordinater

For hvert sjekktidspunkt fra og med `tmin` til og med `tmax`, kjøres en SQL-setning mot datasettet, hvor hver rad nå er på formen <name, timestamp, x, y, geom, group_id>. Kun navn og x,y-koordinater blir hentet der observasjonene samsvarer med søketidspunkt. Disse lagres som lister kalt *names* og X, hvor *names* inneholder alle navn (identifikator) på personene som hadde observasjoner ved søketidspunkt og X er todimensjonal matrise med tilhørende x og y koordinater på alle deltakerne. Denne $n \times 2$ matrisen inneholder allerede interpolerte koordinater på sjekktidspunkt. n er maksimalt lik antall individer, og minimalt lik 1 og varierer mest i starten og mot slutten (på grunn av felles start). Målet med å både ha posisjoner og navn er for å navngi punktene som senere skal inngå i grupper.

Minimum spenntre basert clustering

Cluster-funksjonen tar utgangspunkt i en vektet graf ut fra punktkoordinatene i X. For å konstruere en graf med det valgte networkX-verktøyet, kan en enten legge inn en kant om gangen, eller angi en fullstendig kantliste i en operasjon. Det ble valgt å bestemme kantliste ved hjelp av avstandsmatrise [15, s. 47]. En innebygd funksjon i DistanceMetric-biblioteket, kalt `dist.pairwise` blir brukt til dette med X som input og euklidsk avstand som parameter. Denne funksjonen resturnerer en $n \times n$ matrise med parvise avstander for alle punktene. Et søketidspunktet t_1 kan for eksempel gi

$$X = \begin{bmatrix} x_a & y_a \\ x_b & y_b \\ x_c & y_c \end{bmatrix}$$

hvor (x_a, y_a) er koordinatene til punkt a, (x_b, y_b) er koordinatene til punkt b og tilsvarende for punkt c. Avstandsmatrisen for X blir da

$$D = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 0 & \rho_{ab} & \rho_{ac} \\ \rho_{ba} & 0 & \rho_{bc} \\ \rho_{ca} & \rho_{cb} & 0 \end{bmatrix} \end{array}$$

hvor ρ er euklidsk avstand og diagonalen består avstand til seg selv. Som vi kan se, inneholder D avstander mellom alle parvise kombinasjoner av a,b og c, noe som resulterer i dobbellagring av hver avstand. Hvis man skulle konstruere graf direkte fra D, ville hver kant bli forsøkt lagt inn to ganger (eks. (a,b), (b,a) som er lik), noe som er unødvendig bruk av ressurser. Fordi D er symmetrisk rundt diagonalen, kan en bruke kun øvre eller nedre triangel. Ved bruk av den innebygde Python-funksjon kalt `triu` tas det vare på øvre triangel og resten av matrisen gis 0-verdier. Matrisen blir videre omformet til såkalt CSR[4] representasjon ved hjelp av funksjonen `csr_matrix` fra scipy-biblioteket. Resultatet består av rader på formen [$(rad_i, kolonne_j)$ $verdi$] og utelater 0-verdiene. Tar vi utgangspunkt i D blir avstand a til b [(0, 1) , ρ_{ab}] på CSR-form. Parvise avstander på CSR-formen blir brukt som kantliste for oppretting av vektet graf. Dette gjøres ved kall til funksjonen `from_scipy_sparse_matrix` fra graf-biblioteket networkX. Det resulterende grafobjektet består av kanter med tilhørende vekter (avstander) mellom alle punktpar i X. Alle videre operasjoner på graf-objekt blir også utført ved hjelp av eksisterende funksjoner i networkX-biblioteket.

Neste steg er å bestemme *minimum spenntre*. Med grafobjektet som input, blir det kjørt en funksjon kalt `kruskal_mst`. Den utfører Kruskals algoritme på input-grafen og returnerer *minimum spenntre* som et nytt grafobjekt. Resultatet inneholder alle nodene i X, men kun de kantene som oppfyller kravene for *minimum spenntre* er med. Bestemmelse av *clustere* kan gjøres ved de to kjente tilnærmingene beskrevet i avsnitt 2.2.2. Siden det ikke var kjent på forhånd hvor mange grupper det kunne finnes i datasettet, ble kun avstandskriterie brukt for eliminering av kanter fra det sammenhengende treet. Den gitte verdien `max_dist` i meter angir vekt-grensen for tillatte kanter mellom de nodene som kunne utgjøre et *cluster*. Ved hjelp av traversering av kantlisten i spenntreet kan en bestemme kanter som har vekt større enn eller lik `max_dist`. En midlertidig liste over «ulovlige» kanter blir laget ved å traversere spenntreet, og finne kanter med vekt lik avstandsterskel `max_dist` eller høyere. Listen med eliminerbare kanter blir brukt som input i funksjonen `remove_edges_from`, slik at en får splittet *minimum spanning*-treet i flere subtrær.

De resulterende subtrærne blir hentet ut ved kall på funksjonen `connected_components` og består av flere lister. Hver liste inneholder nummererte navn til de sammenhengende nodene. Enkeltnoder som ikke er forbundet til noen andre inngår også som egne lister. Disse blir filtrert ut ved kontroll på antall elementer `min_size` per liste. Listene består av de nummererte nodene med indekshenvisning i forhold til koordinatmatrise som har lengde lik navnelista `names`. Dette gjør det enkelt å gjøre oppslag på navn til de individene som ble registrert i sammenhengende komponenter. For eksempel kan et *cluster* være en liste på formen [0,1,4], som samsvarer med første, andre og femte element i `names` (som har lengde lik X). Hver operasjon av minimum spenntre *clustering* gir en liste bestående av flere nodelister, en per *cluster*. Etter oppslag i navnelista kommer de på formen `[[a,b],[c,d,e]..]` hvor `[a,b]` er et *cluster*, og `[c,d,e]` er et annet.

Tildeling og lagring av gruppeidentifikator

Hver liste med sammenhengende komponenter blir behandlet for seg. Dette for å holde oversikt over gruppesammensetning med navn på medlemmene. Ideen er å tildele et unikt navn på hver unike gruppesammensetning. Et eksempel er *clusteret* `[a,b,g]`. Dersom eksakt samme *cluster* skulle finnes ved senere kontrolltidspunkt er det viktig at *clusteret* blir gitt samme navn hver gang, som kan tildeles hvert av medlemmene den består av. For å sikre dette, brukes oppslagstabellen som ble opprettet i initialiseringen. Kombinasjoner av medlemsnavn som inngår i et *cluster* blir brukt som nøkkel i denne oppslagstabellen. Verdien tilhørende en nøkkel er numerisk og økende for hver ny nøkkel. Alle navnelister i hvert *cluster* blir sortert alfabetisk, før navnekombinasjonen i form av tupler eks. ('a','b'), blir satt inn som nøkler i oppslagstabellen. Initialverdien blir satt til 1 på aller første registrering av et *cluster*. Neste unike navnesammensetning får id 2 osv. Ved hver kjøring av *clustering*, dvs hver iterasjon sjekkes det om en navnekombinasjon allerede eksisterer i oppslagstabellen og i såfall hvilken gruppeID den har fått

[4] Compressed Sparse Row

tildelt. Gruppeidentifikator blir i samme iterasjon sendt til databasen. Oppdatering av datasettet resulterer i en numerisk verdi for gruppeID-attributt hos de individene som inngikk i grupper ved de aktuelle sjekktidspunktene.

3.2.2. Beregning av gruppenes varighet

Etter at *clustering*-prosedyren er ferdig, er oppgaven å finne varigheten til hvert *cluster* før de eventuelt kan regnes som grupper. Sammenhengende tidsintervaller som representerer «kontinuiteten» til en gruppe kan finnes ved vindusspørringer (*sliding window*) i SQL. I PostgreSQL kalles de *window functions*[5]. Ettersom det oppdaterte datasettet inneholder gruppeID på hver aktuell observasjon samt tidsstempel, kan det både behandles på gruppenivå og medlemmers navn kan hentes ut.

I fremgangsmåten som ble brukt, kjøres to vindusspørringer for å finne sammenhengende tidsperioder innenfor gruppene. Utgangspunktet er kolonnene med gruppeID og tidsstempel. De hentes ut med distinct-operator på gruppeID og sortering på begge kolonnene, slik at hver gruppe får en egen «tidsserie» hvor de har eksistert.

Deretter kjøres første vindusspørring. Den består av et *case-else*-uttrykk for å sjekke om forskjellen mellom tidsstempel i gjeldende og den forrige raden er lik det faste tidsintervallet i data. Hvis dette ikke er tilfellet, markeres raden med 1. Ellers blir denne kontrollverdien satt til NULL i gjeldende rad. Kontrollverdien gis alias *break*. Resultatet av denne vindusspørringen gir avgrensning av de sammenhengende tidsintervallene hvor gruppeID finnes.

En ny vindusspørring gjøres mot resultatet fra forrige spørring. Her summeres *break*-verdiene inkrementelt, slik at summen av 1-ere gjelder for alle rader frem til neste 1-er. Verdien for denne summen gis alias *sequence*. Hver enkelt *sequence*-verdi gjelder uansett gruppeID. Meningen bak *sequence*-verdien er å kunne aggregere på gruppenivå og tidsintervall.

Når en ny spørring gjøres med aggregering på *sequence* og gruppeID, kan det trekkes ut tidspunkter for start- og sluttidspunkt på varigheten til hver av gruppene ved å bruke `min` og *max* funksjon på tidsstempel. Aggregering på *sequence* sikrer at det er sammenhengende tidsintervall mellom de aktuelle tidsgrensene.

Prosedyren med vindusspørringene forklares ved hjelp av en illustrasjon. For eksempelets skyld, antar vi at tabell 3.2 inneholder gruppehistorikken til hele datasettet og fast tidsintervall er 15 sekunder. Videre antas at datasettet består av kolonnene `group_id` og `time` og er allerede hentet ut med sortering på begge. Gangen i vindusspørring 1 er som følger. Tidsstempel i rad 1 sammenlignes mot den i forrige rad. Siden den ikke finnes, settes *break* til 1. De neste fire radene har tidsstempel økende med 15 sekunder i forhold til forrige og settes derfor til NULL (I PostgreSQL vises de som tomme felter). Gruppe 1 og 2 er ferdig behandlet. Når «vinduet» beveger seg videre nedover, finnes første forekomst av gruppe 3. I tidssjekken finnes at differansen i forhold til forrige rad er forskjellig fra 15 sekunder. Det gir en ny *break* representert ved verdien 1. Tilsvarende kontroller og markeringer gjøres til datasettet er ferdig gjennomgått.

Vindusspørring 2 opererer på de resulterende kolonnene, og summerer opp *break*-verdiene på økende tidspunkt og sorterte gruppeID og tidsstempel. Når det tas utgangspunkt i disse *sequence*-verdiene, kan en gjøre `group by` spørring med hensyn på først *sequence*, og deretter gruppeID. Dette gir mulighet til å finne sammenhengende tidsintervall på hver gruppeID. I tabell 3.2 er gruppe 1 og gruppe 2 funnet i sekvens 1. De får en sammenhengende periode hver. Gruppe 5 har to brudd og to sammenhengende perioder. Den første av dem har starttid 8:59:45 og slutt 09:00:15. En trenger kun å ta vare på start- og slittidspunkt for periodene, sammen med gruppeID.

[5] http://www.postgresql.org/docs/9.3/static/functions-window.html

Tabell 3.2.: Bestemmelse av tisdssekvenser med vindusspørring

group_id	time	break	sequence
1	2013-11-02 08:53:15	1	1
1	2013-11-02 08:53:30		1
1	2013-11-02 08:53:45		1
2	2013-11-02 08:54:00		1
2	2013-11-02 08:54:15		1
3	2013-11-02 08:52:00	1	2
3	2013-11-02 08:52:15		2
3	2013-11-02 08:52:30		2
3	2013-11-02 08:52:45		2
3	2013-11-02 08:53:00		2
4	2013-11-02 08:52:00	1	3
4	2013-11-02 08:52:15		3
4	2013-11-02 08:52:30		3
4	2013-11-02 08:52:45		3
4	2013-11-02 08:53:00		3
4	2013-11-02 08:59:30	1	4
5	2013-11-02 08:59:45		4
5	2013-11-02 09:00:00		4
5	2013-11-02 09:00:15		4
5	2013-11-02 09:05:15	1	5
5	2013-11-02 09:05:30		5
5	2013-11-02 09:05:45		5
5	2013-11-02 09:08:30	1	6

En gruppetabell ble opprettet ved hjelp av den beskrevne fremgangsmåten, og fikk kolonnene `group_id`, `tstart`, `t_end`. Den inneholdt alle tidssekvenser for alle de oppdagede gruppene inkludert de som kun gjaldt ett enkelt tidsøyeblikk. I disse tilfellene var det samme tidsstempel for `tstart` og `t_end`.

3.2.3. Filtrering på minimum varighet

Når det først finnes en oversikt over varigheten til gruppene, kan det gjøres enkle spørringer for å trekke ut gruppeID med ønsket minimum varighetsgrense. Dersom en ønsker å hente ut grupper som er blitt registrert på et minimum k antall kontrolltidspunkt på rad, kan det brukes to fremgangsmåter. Det kan kontrolleres om differanse mellom `t_end` og `tstart` dividert med samplingintervall er minst lik k. Fordi periodene for gruppene allerede er i ordnede sekvenser i gruppetabellen, er ikke dette nødvendig. Det enklere alternativet er derfor å sjekke om `t_end-tstart >= min_interval` i en SQL-setning. I PostgreSQL har differanser mellom to tidsstempel datatypen *interval* på formen `'00:01:00'` (ved et minutt). Verdien for `min_interval` må derfor angis som *interval*-type ved f. eks `'00:01:00'::interval`. Ved 10 sekunders samplingsintervall betyr det 10 sammenhengende tidspunkter. Henting av grupper som oppfylte tidskravet ble gjort med denne fremgangsmåten.

For å hente ut antall grupper, ble det gjort en enkel `count` spørring på gruppetabellen på antall unike gruppeID hvor ingen periode var kortere enn minimumskriterie for varighet. Oversikten på gruppenivå med antall ganger en gruppe eksisterte ble fremstilt med utgangspunkt i samme tabell. Her ble det også hentet ut tidspunkter for tidligste og seneste tidspunkt en gruppe hadde blitt identifisert. Når det skulle hentes ut navn på gruppenes medlemmer samt størrelser på gruppene ble gjort en join-spørring på gruppetabell og oppdatert observasjonstabell.

3.2.4. Visualisering

For å produsere en tabell med geometrier på hver enkelt deltaker samt informasjon om gruppetilhørighet ble det også tatt utgangspunkt i gruppetabellen og den oppdaterte observasjonstabellen. En *join*-operasjon krevdes for å linke gruppeinfo til hver enkelt løper basert på gruppeid som fantes i begge tabellene. Linjesegmenter ble produsert slik at de startet når en periode for den respektive gruppe startet og sluttet tilsvarende. Dette ble gjort ved hjelp av en PostGIS-funksjonen `ST_Makeline` som kobler punkter kronologisk etter observasjonstidspunkt og omformer dem til polylinjer. Hver enkelt løper fikk da flere linjegeometrier. Der en person ikke var i noen gruppe var det heller ingen geometri. Det samme gjaldt der det ikke var bevegelse.

Det ble også produsert et datasett med linjegeometrier uavhengig av gruppeattributt med samme funksjon i PostGIS. Et bakgrunnskart med alle individuelle bevegelsesspor var ment til visning av hele historikken, slik at en kunne se både tilhørighet i gruppe og bevegelse individuelt.

Presentasjonene ble laget i programmet QGIS ved å hente geometritabeller inn via databasen. Denne operasjonen krevde bare grunnleggende klassifiseringsverktøy i QGIS. I hele datasettet ble det gjort en klassifisering på attributten gruppeID, slik at felles farge kunne gis til alle som inngikk i samme gruppe.

4. Resultater

Resultatene er produsert ved analyse på et preprosessert datasettet med fast tidsintervall 15 sekunder. I preprosesseringen ble det ikke interpolert mellom observasjonene som hadde tidsgap over 60 sekunder. Dette på grunn av potensiale for feil i antagelsen om rettlinjet bevegelse og konstant hastighet (Se avsnitt 2.1) gitt at det er mennesker som har beveget seg gjennom varierende terreng. Gruppedetektering er testet med flere parametere for avstand og minimum varighet for sjekktidspunkt hvert 15. sekund. Første del av testen tar for seg effekten av de nevnte parameterne på følsomheten til den valgte gruppedetekteringsmetoden. Videre presenteres oversikt over et utvalg av grupper funnet ved to ulike avstandsterskler, samt de individene som inngikk i disse gruppene. Det er også tatt med visualisering av strekninger på de utvalgte gruppene. I alle tilfeller er parameteren for minste gruppestørrelse satt til 2 individer.

4.1. Beskrivelse av data

Datasettet som ble benyttet, stammer fra GNSS-målinger utført høsten 2013 i forbindelse med et orienteringsløp kalt kadaverløpet[1]. Kadaverløpet har vært arrangert årlig av NMBU siden 1958 og foregår i skogområder gjennom Follo i Akershus med løpslengde på ca. 3 mil. Deltakerne blir kjørt til et ukjent sted og må finne veien tilbake til en post på campus, via flere poster underveis. Løpet har felles start, og siste deltaker i mål får tittelen «superkadaver». De siste årene har en gruppe deltakere brukt sporingsenheter i form av utdelte GPS-klokker.

Løpsdata fra kadaverløpet 2013 bestod av 171502 observasjoner totalt, med bevegelsesspor fra 39 løpere. Samplingsraten var variabel, på ca. 2-6 sekunder men flere større gap forekom. Den første observasjonen hadde tidsstempel 08:50:00, og den siste 17:30:14. Det antas at løpsstart var ca klokken 08:50. Løpslengden varierte mellom 32 og 47 kilometer for den enkelte løper. Data var anonymisert, slik at beskrivende attributt i hver observasjon var i form av 'gps01','gps02' osv. Hver av de 171502 observasjonene bestod av en slik identifikator på en løper, tidsstempel, lengde- og breddegrad. Etter preprosesseringen ble datasettet redusert til 51586 observasjoner, med samsvarende tidspunkter i alle bevegelsessporene.

4.2. Test av ulik avstandsterskel og minimum varighet

Analysen er basert på at det ikke finnes a priori kunnskap om hvem som virkelig gikk som grupper eller om forhold på bakken, som sikt, landskapstype osv. Avstandsterskelen max_dist for *clustering*-algoritmen er derfor teoretisk verdi. Det gjelder også kravet om minimum varighet. Klassifisering av grupper ble testet med fire avstander: 10, 25, 50 og 100 meter. Med de valgte avstandene ble en gruppe definert som det *clusteret* på to eller flere punkter som bestod i minimum 1, 2 og 5 minutter.

Dette omfattet kjøring av `mst_cluster_timeseries` i fire runder med ny parameter `max_dist` hver gang. Hver ny kjøring av clustering ble etterfulgt av de window-baserte SQL-setningene på det oppdaterte datasettet. Disse beregnet varigheter av hvert cluster for så å lagre dem i egen gruppetabell, som beskrevet

[1] http://kadavern.oj-oj.net/

i avsnitt 3.2.2. Spørringer på den resulterende gruppetabellen ble gjort for å filtrere ut de gruppene som hadde varighet på minst 1, 2 og 5 minutter. Antall grupper ble funnet ved telling av unike forekomster av gruppeID hos de gyldige gruppene i henhold til varighetskriteriene. Telling av gruppestørrelser ble gjort tilsvarende med *join*-spørring mot gruppetabellen og det oppdaterte datasettet etter *clustering*-prosedyren. For å forenkle videre behandling av gruppedata, ble det tatt kopier av både det oppdaterte datasettet og gruppetabellen etter hver runde. Videre analyse kunne da gjøres direkte i SQL uten å måtte kjøre den iterative clustering-algoritmen flere ganger. Resultatene vises i tabell 4.1.

Tabell 4.1.: Antall grupper funnet ved ulike avstandsterskel og krav om min. varighet

Maks avstand (m)	Minimum varighet	Antall grupper	Største gruppestørrelse
10	1 min.	82	5
	2 min.	39	5
	5 min.	17	5
25	1 min.	112	14
	2 min.	65	8
	5 min.	31	5
50	1 min.	120	27
	2 min.	81	17
	5 min.	46	8
100	1 min.	135	39
	2 min.	93	26
	5 min.	57	10

Det er viktig å presisere at antall grupper betyr antall unike sammensetninger av navn på gruppemedlemmene. Det har vært god tid til endringer i disse med tanke på tidsspennet mellom 08:50 og 17:29. Effekten av dette vises i form av høye tall på antall grupper, selv om data er fra 39 deltakere totalt. Tabell 4.1 viser også hvor mange medlemmer det var i de største gruppene med hensyn på de ulike avstands- og varighetskriteriene.

Store grupper forekommer mest ved stor tillatt maksavstand og kort minimum varighet. Det er som forventet med tanke på mulige clustere som kan bestemmes med minimum spenntre clustering. Punktene i slike clustere kan være koblet via felles naboer og ha vilkårlig utstrekning. Store grupper kan samtidig indikere den tidsvise romlige fordelingen i data.

Resultatene fra denne testen ble brukt som utgangspunkt for videre analyse på gruppenivå og hvem gruppene bestod av. Det ble besluttet å følge opp resultatene funnet ved avstandskriteriene 25 og 50 meter og minste varighet 5 minutter. Med 15 sekunders samplingsrate betyr dette gjentatte klassifiseringer av samme clustere i 20 etterfølgende sjekktidspunkt. Ingen nabopar i disse gruppene kunne være lenger unna enn 25 og 50 meter henholdsvis (antatt mulighet for visuell kontakt).

4.3. Oversikt over grupper og gruppemedlemmer

Når gruppene var funnet etter kriteriene beskrevet over, var neste steg å finne ut hvor mange ganger de ble detektert over det totale tidsspennet i data. Det var også av interesse å sjekke hvor lenge gruppene eksisterte totalt, når de ble oppdaget først, og når de sist gjaldt som grupper. Denne informasjonen ble fremstilt ved spørring på gruppetabellen som inneholdt sammenhengende perioder og start- og sluttidspunkt for disse periodene. Gruppene som varte over fem minutter ble hentet ut, og enkle SQL-funksjoner ga de ønskede resultatene: gruppeID, total varighet, lengste periode og tidspunkter for første og siste registrering. På

grunn av stor datamengde i resultatene er det kun vist resultater for de ti lengstvarende gruppene. Tabell 4.2 inneholder resultatene for grupper funnet ved `max_dist` 25 meter. Tabell 4.3 består av tilsvarende for grupper identifisert når `max_dist` var 50 meter. Navn på gruppemedlemmene tilhørende disse resultatsettene vises i tabellene 4.4 og 4.3.

I tabellene 4.2 og 4.3 er antall perioder antall sammenhengende tidsperioder på minst 5 minutter. Det innebærer clustere som har bestått kontinuerlig i 5 minutter når det ble sjekket hvert 15.sekund. Total varighet er angitt i minutter, og står for summen av alle disse kontinuerlige periodene. Lengste perioder er også tatt med.

Tabell 4.2.: Ti lengstvarende grupper funnet ved avstandsterskel 25 meter

GruppeID	Ant. perioder (≥5 min.)	Varighet tot. (minutter)	Lengste periode (minutter)	Første periodestart (klokkeslett)	Siste periodeslutt (klokkeslett)
16	17	443.00	131	09:04:30	17:29:45
55	11	275.50	115	09:29:45	15:36:15
39	19	348.75	76	08:55:30	16:46:30
83	12	163.00	32	09:05:30	14:24:15
127	7	106.50	32	09:23:15	12:59:15
6	1	22.50	22	09:02:15	09:24:45
24	6	98.00	22	10:10:45	13:04:15
178	5	63.25	21	09:37:15	12:04:00
25	6	46.25	18	08:54:45	14:06:45
63	6	61.50	18	09:20:45	10:47:00

Tabell 4.3.: Ti lengstvarende grupper funnet ved avstandsterskel 50 meter

GruppeID	Ant. perioder (≥5 min.)	Varighet tot. (minutter)	Lengste periode (minutter)	Første periodestart (klokkeslett)	Siste periodeslutt (klokkeslett)
28	12	445.00	122	09:04:45	17:29:45
30	9	271.00	116	09:30:00	15:36:15
20	10	377.75	78	09:49:15	16:46:30
73	4	134.25	65	09:24:45	12:58:45
137	17	295.25	38	09:54:00	16:01:30
29	10	133.25	35	09:03:00	14:08:15
55	11	156.75	34	09:06:30	14:24:15
14	4	104.00	32	08:56:15	10:49:45
253	1	31.75	31	13:07:30	13:39:15
209	3	50.25	30	12:12:00	13:09:00

Gitt at avstandskriterium er realistisk, kan de lengste periodene angi de gruppene som fantes i virkeligheten. For eksempel har gruppe 16 i tabell 4.2 vart så godt som i hele løpet med tanke på summen av tidsperiodene de ble regnet som grupper. Denne gruppen består av to medlemmer (se tabell 4.4). Det høye antallet perioder for denne gruppen indikerer også mange «brudd», dvs. 16 tilfeller.

Ved avstandsterskel på 50 meter ble den også den samme gruppen lengstvarende (nå med gruppeID 28). Antall perioder er i dette tilfellet er lavere, hvilket også betyr færre «brudd». Deres lengste sammenhengende periode er registrert som kortere ved avstandsterskel 50 meter i forhold til den funnet ved 25 meter. Det betyr at de to gruppemedlemmene har holdt seg innenfor 25 meters avstand til hverandre i 131 minutter, men at avstandsgrensen 50 meter har inkludert andre individer. Dette kommer av at regelen for tildeling av gruppeID er basert på fast gruppesammensetning. Et eksempel på denne effekten vises i tabell 4.4 (b) og 4.3 hos gruppe 55 og 253. Gruppe 55 fikk selskap av et nytt individ over en halv time sammenhengende (og var da omdøpt til gruppe 253).

Tabell 4.4.: Gruppemedlemmer ved (a) 25 og (b) 50 meter avstandsterskel

(a)

GruppeID	Medlemmer
16	gps05 gps10
55	gps06 gps15
39	gps16 gps18
83	gps41 gps47
127	gps46 gps63
6	gps03 gps44
24	gps19 gps58
178	gps44 gps62
63	gps48 gps56
25	gps66 gps67

(b)

GruppeID	Medlemmer
28	gps05 gps10
30	gps06 gps15
20	gps16 gps18
73	gps46 gps63
137	gps12 gps64 gps68
29	gps66 gps67
55	gps41 gps47
14	gps48 gps56
253	gps03 gps41 gps47
209	gps04 gps52

At det er forskjellig gruppeID på samme gruppe kommer av at clustering-algoritmen er kjørt flere ganger med ulik avstandskriterie hver gang. Som forklart i 3.2 tildeles gruppeID på tidsøyeblikk (før varigheten beregnes) når clustere er funnet.

4.4. Visualisering av strekninger

Neste del av oppgaven var å studere de sammenhengende strekningene visuelt på de ti gruppene funnet tidligere. Basert på gruppetabellen og det oppdaterte datasettet med gruppeID, ble det generert en ny tabell for visualisering. Denne tabellen ble importert i QGIS, og gruppenes geometri ble splittet, slik at det ble ett kartlag for hver gruppe. Strekningene for gruppene fremstilt ved max_dist 25 meter er vist i figur 4.1. En tilsvarende visualisering for grupper funnet ved max_dist 50 meter er i figur 4.2.

Visualiseringene er et supplement til tabellene 4.2 og 4.3 og 4.4. Kartene består av linjegeometrier for alle enkeltindividene som deltok i kadaverløpet 2013 som bakgrunnskart i grått. Geometriene for gruppene angir kun der det var bevegelse og består av individuelle polylinjer for de enkelte gruppemedlemmene. De er uthevet og gitt felles farge for å angi en aggregert visning.

Resultater for de tre lengstvarende gruppene er nærmest identiske på kartene som gjelder begge avstandstersklene. For gruppe 55, som en periode ble til gruppe 253, kan strekningen med utvidet gruppe sees i figur 4.2. En kan også se i samme figur at i den siste strekningen bestod gruppen av to personer. Tilsvarende brudd i strekningene forekommer hos andre grupper også. Det kan bety enten at gruppesammensetningen ble endret, eller at gruppemedlemmene var for langt unna hverandre til å kunne regnes som grupper.

Det er også viktig å merke seg at antall perioder i tabellene 4.2 og 4.3 ikke alltid stemmer med strekningene vist i kartene. Dette kommer av at strekningene på kartene kun omfatter bevegelse. Som nevnt i 1.2.4, produseres det ikke linjegeometrier der personer har stått i ro.

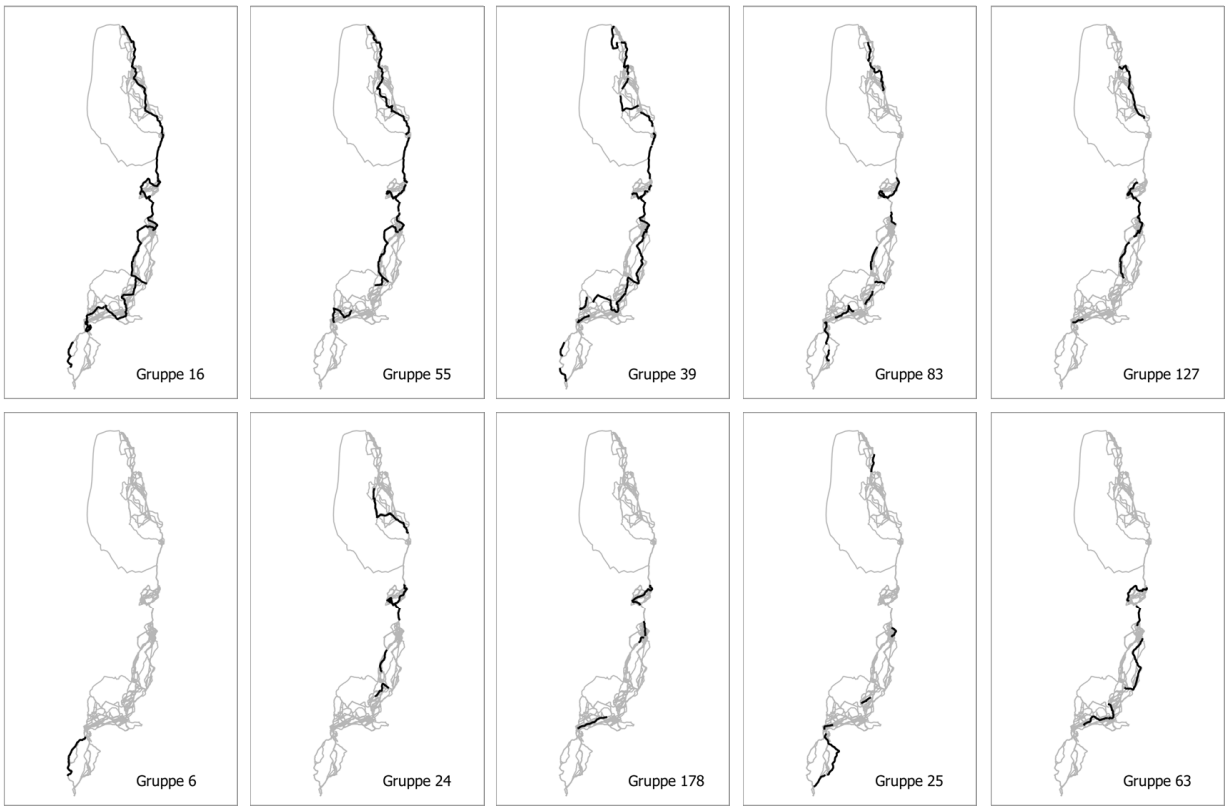


Fig. 4.1.: Grupper med lengst varighet ved 25 meter avstandsterskel

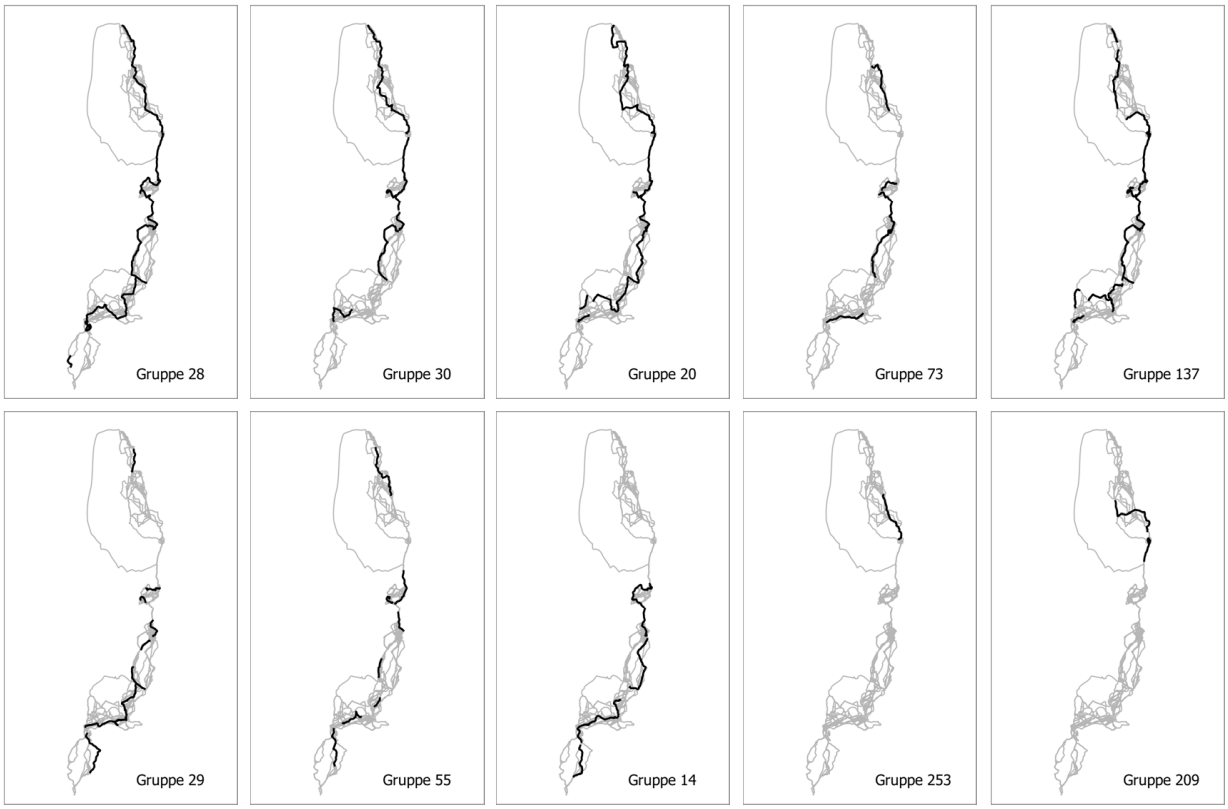


Fig. 4.2.: Grupper med lengst varighet ved 50 meter avstandsterskel

5. Diskusjon

Resultatene fra testen av avstands- og varighetsparametere i kap. 4.2 oppsummerer på flere måter effekten av forenklinger og antagelser i gruppedetekteringsmetoden. For eksempel har 100 meter avstandsterskel regnet alle de 39 deltakerne som en gruppe i mellom ett og to minutter. Fordi løpet hadde felles start, er ikke dette overraskende. En har lagt til grunn at posisjonshistorikk fra orienteringsløpet alene kan brukes til å finne mulige grupper blant deltakerne. Problemet har da blitt redusert til å finne sett av punkter som har vært i hverandres nabolag over en viss tid [4, 10]. En viktig vurdering som må gjøres i slike tilfeller, er å velge teknikk for bestemmelse av nabolag som er mest mulig realistisk. Minimum spenntre-*clustering* ble benyttet for å inkludere de som kan ha visuell kontakt, men også de som ikke nødvendigvis er direkte naboer med hverandre men via andre. Splitting av minimum spenntreet er gjort med det enkleste prinsippet, dvs eliminering av kanter hvor vekt (som representerer avstand i vårt tilfelle) er høyere enn en spesifisert verdi. Parameteren for maks innbyrdes avstand tillatt mellom de som er i gruppe er avgjørende for hvor streng denne *clustering*-metoden blir. Dette igjen vil gi utslag på hvor mange og hvem som regnes som grupper samt de som utelates. Som følge av konnektivitetsegenskapen til minste spenntre, er det en fare for overestimering i gruppebestemmelse. Kontinuitetstesten med minimum varighet er det som skiller en gruppe i rom-tid fra et statisk *cluster*. Dette kan utelukke tilfeldige passeringer. En ser likevel at på den største tillatte avstanden og lengste minimum varighet (tabell 4.1) en mulig overestimering. Det kan komme av at flere små grupper sammen har blitt regnet som en større gruppe på 10 personer. I den andre enden (10 meter maks avstand) blir ingen grupper større enn på 5 individer gjennom alle de tre varighetskriteriene. Fordi data angår orienteringsløp, kan en regne små grupper som mer sannsynlige enn de store. Formålet med avstandstesten var både å se hvordan metoden oppfører seg med ulike parametre, men også om romlig fordeling i data. Siden det ikke fantes informasjon om hvem som virkelig gikk sammen, enten ved at de samarbeidet eller fulgte andre i mangel på egne orienteringsferdigheter er det beste man har å gå etter en tallfestet mulighet for at dette var tilfelle. Iterativ *clustering* med minimum spenntre og etterfølgende kontroll på tidsserien viste at en kan skille sannsynlige grupper(på to eller flere individer) fra de isolerte individene. Dette i form av avgrensning på både aktuelle tidsperioder og gruppeID-attributt som anga personers tilhørighet til deres respektive grupper. Til tross for svakheten med overestimering, er det viktig å merke seg at metoden gir konsistente resultater (som avstands- og kontinuitetstesten viser).

Nærmere kontroll på de lengstvarende gruppene viser mye dynamikk med tanke på hvor mange enkeltperioder de ble regnet som grupper. Dette gir indirekte informasjon om mulige brudd og overganger. Ideelt sett bør det være en sammenhengende periode over hele tidsløpet. Metoden er noe naiv i bestemmelse av gruppemedlemskap og fanger derfor ikke opp endringer eksplisitt i gruppesammensetning. Et eksempel ble gitt i 4.3, hvor samme gruppe ble regnet som en ny gruppe når det kom et nytt medlem til. Tildeling av gruppeidentifikator går ut fra at en fast navneliste etter *clusteringen*, og den må holde seg konstant over en viss tid før en kan angi en aktuell gruppe. Den intuitive forståelsen av grupper samsvarer ikke nødvendigvis med denne fasen i fremgangsmåten, da grupper kan vokse og krympe og fortsatt ha noen faste medlemmer. Output gitt etter kjøring av metoden og etterbehandling i SQL, gir likevel muligheter for å finne overlappende gruppemedlemskap. Dette fordi gruppeID blir lagret som attributt i enkeltobservasjoner. Bestemmelse av gruppeID etter den nevnte fremgangsmåten kan i tillegg til å finne mulige grupper gi svar på: (1) Hvilke individer har inngått i grupper (2) Hvor lenge de har holdt seg nærme hverandre og når dette var tilfelle.

Fra GIS-perspektiv er det også interessant å undersøke den visuelle fremstillingen av en gruppeklassifisering.

Kartene i for eksempel figur 4.1 viser et helhetlig bilde av grupper i bevegelse ved enkle kartografiske midler. En sannsynlig gruppe bør bestå i rom og tid. Kombinert med oversikt over periodene en gruppe har eksistert, gir tidsinformasjonen et mye rikere bilde. Dette fordi tidsgrenser i periodene er knyttet til geometriene der det har skjedd bevegelse. Et godt eksempel er den ene gyldige perioden for gruppe 6 i tabell 4.2. Kartet viser denne strekningen, slik at en vet ut fra beregningene at det var bevegelse hos denne gruppen i over 22 minutter og hvor den aktuelle strekningen er på kartet. Geometridatasettet er laget på en slik måte at det tas vare på start- og sluttidspunkter tilhørende hver polylinje. Dette åpner for mange flere muligheter enn de som ble presentert i forrige kapittel.

Ved nabobestemmelse i rom-tid data er det to viktigte faktorene å ta hensyn til: usikkerheten i observasjonsteknikk og konteksten begelsen skjer i. Selv med stor nøyaktighet i observasjonene (som er mulig med GNSS), bør en ta hensyn til at bevegelsestype hos mennesker kan variere etter situasjon. En liten gruppe med venner kan typisk gå ved siden av hverandre, mens en større gruppe kan bestå av mindre «interne» grupper. Metoden som er brukt, kan potensielt finne slike grupper når det finnes detaljert posisjonshistorikk i form av bevegelsesspor.

Videre arbeid kan inkludere flere av elementene nevnt over. For eksempel muligheter for å redusere antall grupper ved å forbedre rutinen for tildeling av gruppeID. Det kan undersøkes nærmere hvordan overganger, oppsplittinger og brudd kan fanges opp på en bedre måte. Algoritmen er i stor grad SQL-basert, og resultatdatasettet kan behandles videre i databasesystemet PostgreSQL. Spørringer på delmengder kan brukes til å finne for eksempel når en gruppe ble større eller mindre.

Et annet viktig element for mulig videre arbeid, er hvor mye samplingsraten har å si for kontinuiteten til grupper funnet med *clustering* på tidsøyeblikk. I denne oppgaven ble en fast samplingsrate på 15 sekunder valgt med antagelsen om at få dramatiske endringer kan skje på denne tiden. Det kunne være interessant å se akkurat hvor raske endringer i posisjon, og dermed potensiell gruppemedlemskap kan være i tilsvarende data. Dette vil innebære kun få endringer i forhold til parametere valgt i denne oppgaven.

Clustering-metoden DBSCAN ble nevnt i teorikapittelet, selv om den ikke ble anvendt i metoden. En kunne få tilsvarende resultater som de presentert tidligere, dersom en valgte *minPts* lik 2 og e lik avstandsterskel brukt i oppgaven. Styrken til DBSCAN kan derimot vises når minimum antall naboer er større enn to. Det ville derfor være en interessant med en sammenlignende analyse av DBSCAN og minimum spenntre i klassifisering av grupper. I såfall vil en kunne tilpasse algoritmen som er implementert for dette prosjektet. Det er støtte for DBSCAN i tilleggsbiblioteket `scikit-learn`, som ble benyttet for avstandsberegninger i forbindelse med oppretting av graf.

6. Konklusjoner

I dette prosjektet ble det implementert og testet en generalisert metode for å finne grupper blant folk ut fra deres registrerte bevegelsesspor. Metoden for å bestemme potensielle grupper i enkelttidspunkter og sjekke deres kontinutitet, tillater stor romlig spredning innen de enkelte gruppene. Samtidig var ikke datasettet et intuitivt valg for kontroll av denne type metoder. Det bestod av GNSS-spor fra et orienteringsløp med fellesstart. En kunne derfor ikke vente stor spredning hos de sannsynlige gruppene. Det ble antatt at en gruppe består av to eller flere personer som holder seg i nærheten av hverandre, enten ved at alle har visuell kontakt med hverandre (ved gruppe på to) eller via andre i mellom (ved større grupper). De ble regnet som grupper dersom deres naboskap varte minst 5 minutter.

Det ble funnet at den minimum spenntre-baserte metoden kan isolere potensielle grupper selv på et datasett med liten romlig spredning og felles start i bevegelsessporene. En må imidlertid være forsiktig ved valg av parametere for nabobestemmelse, på grunn av risiko for overestimering. Denne vurderingen er dessverre vanskelig i praksis, og er avhengig av bruksområde. Selv innenfor samme kontekst kan det være ulike bevegelsesmønstre hos mennesker, særlig når det gjelder rekreasjon og fritidsaktivitet. Hvis en ønsker å finne en gruppe av folk som er betydelig langt unna andre mennesker, kan denne metoden være nyttig. Slike problemstillinger kan finnes i situasjoner hvor det er viktig at folk holder seg nærme hverandre, for eksempel ekspedisjoner i fjerne og vanskelige terreng. I slike tilfeller kan gruppedetektering også brukes på invers form, dvs for å påvise brudd hos de registrerte gruppene. I tillegg til å finne mulige grupper, kan en med den beskrevne metoden produsere oversikt over hvem som har gått eller reist sammen. Nytteverdien kan være enda større hvis en tilsvarende metode tilpasses til bruk i sanntid.

I dette prosjektet ble det ikke tatt hensyn til usikkerhet og avvik i målingene, selv om de potensielt er store i GNSS-observasjoner. Fokuset var å utforske en klassifiseringsteknikk som opererer på rom-tid punkter. Det viktige var derfor at metoden gir konsistente resultater.

Referanser

[1] Andrienko, G. , Andrienko, N. , Bak, P. , Keim, D. , Kisilevich, S. , and Wrobel, S. . A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages & Computing*, 22(3):213–232, 2011.

[2] Andrienko, N. and Andrienko, G. . Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization*, page 1473871612457601, 2012.

[3] Andrienko, N. , Andrienko, G. , Pelekis, N. , and Spaccapietra, S. . Basic concepts of movement data. In *Mobility, data mining and privacy*, pages 15–38. Springer, 2008.

[4] Benkert, M. , Gudmundsson, J. , Hübner, F. , and Wolle, T. . Reporting flock patterns. *Computational Geometry*, 41(3):111 – 125, 2008. ISSN 0925-7721. doi: http://dx.doi.org/10.1016/j.comgeo.2007.10.003. URL http://www.sciencedirect.com/science/article/pii/S092577210700106X.

[5] Djuknic, G. and Richton, R. . Geolocation and assisted gps. *Computer*, 34(2):123–125, Feb 2001. ISSN 0018-9162. doi: 10.1109/2.901174.

[6] Dodge, S. , Weibel, R. , and Lautenschütz, A.-K. . Towards a taxonomy of movement patterns. *Information visualization*, 7(3-4):240–252, 2008.

[7] Ester, M. , Kriegel, H.-P. , Sander, J. , and Xu, X. . A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[8] Giannotti, F. , Nanni, M. , Pedreschi, D. , Pinelli, F. , Renso, C. , Rinzivillo, S. , and Trasarti, R. . Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB Journal—The International Journal on Very Large Data Bases*, 20(5):695–719, 2011.

[9] Grygorash, O. , Zhou, Y. , and Jorgensen, Z. . Minimum spanning tree based clustering algorithms. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 73–81. IEEE, 2006.

[10] Jeung, H. , Yiu, M. L. , Zhou, X. , Jensen, C. S. , and Shen, H. T. . Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[11] Kreyszig, E. . *Advanced engineering mathematics*. Wiley, 2011.

[12] Laube, P. , Kreveld, M. , van, and Imfeld, S. . Finding remo—detecting relative motion patterns in geospatial lifelines. In *Developments in spatial data handling*, pages 201–215. Springer, 2005.

[13] Long, J. A. and Nelson, T. A. . A review of quantitative methods for movement data. *International Journal of Geographical Information Science*, 27(2):292–318, 2013. doi: 10.1080/13658816.2012.682578. URL http://dx.doi.org/10.1080/13658816.2012.682578.

[14] Orellana, D. , Bregt, A. K. , Ligtenberg, A. , and Wachowicz, M. . Exploring visitor movement patterns in natural recreational areas. *Tourism Management*, 33(3):672–682, 2012.

[15] O'Sullivan, D. and Unwin, D. . *Geographic information analysis*. John Wiley & Sons, 2014.

[16] Pelekis, N. and Theodoridis, Y. . *Mobility Data Management and Exploration*. Springer-Verlag New York, 2014. ISBN 978-1-4939-0391-7.

[17] Seeber, G. . *Satellite geodesy: foundations, methods, and applications.* Walter de Gruyter, 2003.

[18] Takenga, C. and Kyamakya, K. . A low-cost fingerprint positioning system in cellular networks. In *Communications and Networking in China, 2007. CHINACOM'07. Second International Conference on*, pages 915–920. IEEE, 2007.

[19] Topographix. The GPS Exchange Format. http://www.topografix.com/gpx.asp, 2016. [Online; Besøkt 22.jan 2016].

[20] Vathy-Fogarassy, Á. and Abonyi, J. . *Graph-Based Clustering and Data Visualization Algorithms*, chapter Graph-Based Clustering Algorithms, pages 17–41. Springer London, London, 2013. ISBN 978-1-4471-5158-6. doi: 10.1007/978-1-4471-5158-6_2. URL http://dx.doi.org/10.1007/978-1-4471-5158-6_2.

[21] Zahn, C. T. . Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, 100(1):68–86, 1971.

[22] Zogg, J. . Gps–essentials of satellite navigation. *Compendium*, 58, 2009. URL https://www.u-blox.com/en/technology/GPS-X-02007.pdf. [Online; Besøkt 2.mai 2016].

Tillegg A.

Python filer

A.1. Preprosesseringsrutine

```
# -*- coding: utf-8 -*-

import numpy as np
import handle_db as db
import time_conv as tc

def interpolate(t, t_prv, t_nxt, x_prv, y_prv, x_nxt, y_nxt):
    # spesialtilfelle: ingen bevegelse
    overlap = t_nxt - t_prv == 0
    if overlap:
        # Interpolerer ikke
        x,y = x_prv,y_prv
    else:
        x = x_prv + (t-t_prv)*((x_nxt-x_prv)/(t_nxt-t_prv))
        y = y_prv + (t-t_prv)*((y_nxt-y_prv)/(t_nxt-t_prv))
    return x,y


def process(ti,table,new_table,id_col,t_col,x_col,y_col):
    t_srch = tc.sec2timestamp(ti)
    Xi = db.sql_nearest_t(t_srch,table,id_col,t_col,x_col,y_col)
    for row in Xi:
        if None not in row:
            idc,t_prv,t_nxt,x_prv,x_nxt,y_prv,y_nxt=row
            if t_nxt-t_prv>=max_gap: #maksimalt gap
                pass
            else:
                x,y= interpolate(ti, t_prv, t_nxt, x_prv, y_prv, x_nxt, y_nxt)
                rec = (idc,t_srch,x,y)
                db.sql_populate_tbl(new_table,id_col,t_col,x_col,y_col,rec)


#tidsintervall sek.
tdelta = 15

#Maks tillatt gap for interpolering
max_gap = 60
```

```
# database param
table = 'kadaver13_utm32'
dbname = 'orientering'
host = 'localhost'
port = 5432
user = 'postgres'
passwd = 'pass'

# kolonnenavn i tabell
id_col='unit'
t_col='time'
x_col='x'
y_col='y'
srid=32632

db.conn = db.connect_to_pgDb(dbname,host,port,user,passwd)
db.cursor = db.conn.cursor()


#opprett ny tabell med interp.data
new_table = table+'_interp_'+str(tdelta)+'s'
db.create_pgTable(new_table,id_col,t_col,x_col,y_col,srid)

#tstart, tslutt
tmin,tmax=db.get_tlimits(table,t_col)

print('processing {} with time interval {}'.format(table,tdelta))
for t in np.arange(tmin,tmax,tdelta):
    process(t,table,new_table,id_col,t_col,x_col,y_col)

db.set_geomSRID(new_table,x_col,y_col,srid)

# avslutt db forbindelse.
db.db_close()

print('created new table {}'.format(new_table))
```

A.2. MST cluster timeseries

```
# -*- coding: utf-8 -*-

import handle_db as db
import time_conv as tc
import numpy as np
from scipy import sparse
import networkx as nx
from sklearn.neighbors import DistanceMetric
```

```
def get_obs(ts):
    expr = """ select {},{},{} from {}
    where time= '{}';"""
    expr = expr.format(name_col,x_col,y_col,tablename,ts)
    recs = db.sql_select_query(expr)
    Xi = [list([a,np.float(b),np.float(c)]) for (a,b,c) in recs]
    Xi = np.array(Xi)
    return Xi


def send_to_db(ts,gr_id,names):
    expr = """update {table} set group_id = {gr_id}
    where {time} = '{ts}' and {name} in {names}"""
    expr=expr.format(table=tablename,gr_id=gr_id,time=t_col,ts=ts,name=name_col,
                     names=names)
    print(expr)
    db.sql_execute(expr)


def mst_clus(X,max_dist):
    #Avstandsmatrise
    dist = DistanceMetric.get_metric('euclidean')
    data = dist.pairwise(X)
    upperTr = np.triu(data,0)
    XX= sparse.csr_matrix(upperTr)
    # Opprett graf og MST
    G =nx.from_scipy_sparse_matrix(XX)
    T = nx.kruskal_mst(G)
    # kutt MST-kanter over max_dist
    T.remove_edges_from((u,v) for u,v,d in T.edges_iter(data=True)
    if d['weight']>max_dist)
    subgroup_list=nx.connected_components(T)
    clus = [g for g in subgroup_list if len(g)>=2]
    return clus


def assign_groupID(ts,cluslist,names):
    for g in sorted(cluslist):
        c = tuple(sorted(names[g]))
        if not(assigned):
            gr_id = 1
            assigned[c] = gr_id
        elif assigned.has_key(c):
            gr_id = assigned[c]
        else:
            gr_id = max(assigned.values())+1
            assigned[c] = gr_id
        send_to_db(ts,gr_id,c)


def mst_cluster_timeseries(t):
    ts = tc.sec2timestamp(t)
    Xi = get_obs(ts)
```

```
    if None in Xi: #tomme rader
        pass
    else:
        names,x,y = Xi[:,0],Xi[:,1],Xi[:,2]
        print(names)
        X = np.vstack((x, y)).T
        clist = mst_clus(X,max_dist)
        assign_groupID(ts,clist,names)


tablename = 'kadaver13_utm32_interp_15s'
dbname = 'orientering'
host = 'localhost'
port = 5432
user = 'postgres'
passwd = 'pass'

# kolonnenavn i tabell
name_col='unit'
t_col='time'
x_col='x'
y_col='y'


db.conn = db.connect_to_pgDb(dbname,host,port,user,passwd)
db.cursor = db.conn.cursor()

tmin,tmax=db.get_tlimits(tablename,t_col)

assigned = {}
gr_id=None
t_delta = 15
#max_dist = 100
#max_dist = 50
#max_dist = 25
max_dist = 100

modify_tbl_x= """alter table {tbl} drop column if exists group_id;
             alter table {tbl} add column group_id integer"""
expr = modify_tbl_x.format(tbl=tablename)
db.sql_execute(expr)
db.db_commit()

for t in np.arange(tmin,tmax,t_delta):
    mst_cluster_timeseries(t)

db.db_close()
```

A.3. Hjelpemoduler

```
# time_conv.py
import datetime
import time

def datetime2sec(d_time):
    """
    Konverter datoobjekt datetime til antall sekunder
    siden epoke 1.jan.1970 00:00:00  """
    return (d_time-datetime.datetime(1970,1,1)).total_seconds()


def sec2timestamp(total_sec):
    """
    Konverter sekunder (siden 1.1.1970 00:00:00) til datoobjekt
    output er datostreng i formatet YYYY-mm-dd HH:MM:ss"""
    return time.strftime('%Y-%m-%d %H:%M:%S', time.gmtime(total_sec))
```

```
# handle_db.py
import psycopg2 as pg
def connect_to_pgDb(_dbname,_host,_port,_user,_passwd):
    """-------------------------------------------------------------------
    Koble til PostGreSQL database. Returnerer connection-objekt conn
    """
    try:
        conn = pg.connect(dbname=_dbname, host=_host, port=_port, user=_user,
                                password=_passwd)
    except:
        print('Kunne ikke koble til databasen')
    else:
        return conn
def db_commit():
    conn.commit()

def db_close():
    conn.commit()
    conn.close()
    cursor.close()

def sql_select_query(expr):
    cursor.execute(expr)
    Xi=cursor.fetchall()
    #print(Xi)
    return Xi

def sql_execute(expr):
    cursor.execute(expr)

def sql_populate_tbl(table,id_col,t_col,x_col,y_col,rec):
    expr = """insert into {table} ({id_col},{t},{x},{y})
    values {rec}""".format(table=table,id_col=id_col,t=t_col,x=x_col,y=y_col,
    rec=rec)
    print(expr)
    sql_execute(expr)

def get_tlimits(table,time_col):
    u"""-------------------------------------------------------------------
    Finner tidligste og seneste observerte tidspunkt i SQL tabell.
    Returnerer t_start,t_end som antall sekunder siden epoke 1.1.1970 00:00:00.
    """
    sql_str = """ select extract (epoch from min({_time_})),
    extract (epoch from max({_time_}))
                from {table} limit 1 """.format(_time_=time_col,table=table)

    cursor.execute(sql_str)
    t_start,t_end = cursor.fetchall()[0]
    return t_start,t_end

def sql_nearest_t(t_srch,table,id_col,time_col,x_col,y_col):
    u""" Finner to sett med obs ut fra gitt tidspunkt. Ett for tidspunkt foer
    og ett med tidspunkt etter sjekktidspunkt t_srch"""
```

```
    query = """select a.{_id_},extract (epoch from a.tid),
    extract (epoch from b.tid),a.{x},b.{x},a.{y},b.{y} from
    (select distinct gps1.{_id_}, gps1.{_time_} as tid,
            {x}, {y}
            from {table} gps1,
                 (select gps2.{_id_}, max(gps2.{_time_}) as tid
                    from {table} gps2
                    where {_time_} <= '{t_srch}Z'
                    group by gps2.{_id_}) as temp
                where gps1.{_id_} = temp.{_id_}
                and gps1.{_time_} = temp.tid
                order by gps1.{_id_}) as a
                left join
    (select distinct gps1.{_id_}, gps1.{_time_} as tid,
            {x}, {y}
            from {table} gps1,
                 (select gps2.{_id_}, min(gps2.{_time_}) as tid
                    from {table} gps2
                    where {_time_} >= '{t_srch}Z'
                    group by gps2.{_id_}) as temp
                where gps1.{_id_} = temp.{_id_}
                and gps1.{_time_} = temp.tid
                order by gps1.{_id_}) as b
                on a.{_id_}=b.{_id_};"""

    expr = query.format(t_srch=t_srch,_id_=id_col,_time_=time_col,x=x_col,
    y=y_col,table=table)
    Xi = sql_select_query(expr)
    return Xi

def create_pgTable(tablename,id_col,t_col,x_col,y_col,srid):
    expr = """drop table if exists {table} cascade;
    create table {table}(
          gid serial not null,
          {_id_} varchar(10),
          {ts} timestamp,
          {x} numeric,
          {y} numeric,
          geom geometry(Point,{srid}),
          constraint {table}_pkey primary key(gid)
          );"""
    expr=expr.format(table=tablename,_id_=id_col,ts=t_col,x=x_col,y=y_col,
                     srid=srid)
    sql_execute(expr)

def set_geomSRID(table,x_col,y_col,srid):
    expr="""update {table}
    set geom = ST_SetSRID(ST_MakePoint({x},{y}),{srid})"""
    expr = expr.format(table=table,x=x_col,y=y_col,srid=srid)
    sql_execute(expr)

# initialverdi
```

```
conn=0
cursor=0
```

Tillegg B.

Etterbehandling i SQL

B.1. Tabell for gruppenes perioder

```
-- window basert sporring. ny tabell med sammenhengende perioder for grupper
drop table if exists groups;
create table groups as
select min(group_id) as group_id, min(time) as tstart, max(time) as t_end
from
(select *, sum(break) over (order by group_id,time) as seq from
(select group_id,time,
        case
                when time - lag(time) over (order by group_id) ='00:00:15'
                then null
                else 1
        end as break
from
(select distinct group_id,time from
kadaver13_utm32_interp_15s
order  by group_id,time) t1
)t2
)t3
group by seq,group_id
order by tstart,group_id

--- Antall grupper, min varighet 5 min
select count(distinct group_id)
from groups
where group_id is not NULL
and t_end-tstart >= '00:05:00';

-- max gruppestr, min varighet 5 min
select max(str) from (
select group_id, count(distinct unit) as str
from kadaver13_utm32_interp_15s
where group_id in
(select distinct group_id
from groups
where group_id is not NULL
and t_end-tstart >= '00:05:00'
)
```

```
group by group_id
) t
```

B.2. Statistikk på grupper med lengst varighet

```
create view longest_group_periods as
(select distinct(group_id), count(*) as instances,
round(sum(extract(epoch from t_end-tstart))::numeric/60,2)
as total_duration,
max(round(extract(epoch from t_end-tstart)::numeric/60,2))
as max_duration,
to_char(min(tstart) ,'HH24:MI:SS') as first_obs,
to_char(max(t_end) ,'HH24:MI:SS') as last_obs
from(
select * from groups
where t_end-tstart >= '00:05:00'
and group_id is not null
) t
group by group_id
order by max_duration desc
limit 10
)
```

B.3. Oversikt over gruppemedlemmer

```
-- grupper med lengst varighet
select group_id,names from (
select distinct(a.group_id), string_agg(distinct(unit),' ') as names,
b.max_duration as ord
from kadaver13_utm32_interp_15s a,longest_group_periods b
where a.group_id = b.group_id
group by a.group_id,
b.max_duration) t
order by ord desc
```

B.4. Linjegeometrier for visualisering

```
-- linjegeometrier for gruppemedl.
drop table if exists traj_grp;
create table traj_grp as
select unit,group_id,tstart,t_end, st_makeline(geom order by time) as geom
from (
select t.unit,t.time,t.geom,t.group_id as gr_id,g.group_id,g.tstart,g.t_end
from kadaver13_utm32_interp_15s t, groups g
where t.group_id = g.group_id
and t.time between g.tstart and g.t_end
```

```
and g.t_end-g.tstart>='00:05:00') s
group by unit,group_id,tstart,t_end

-- kun de ti lengste gruppene

create table visualise as
select * from traj_grp
where group_id in
(
select group_id from
longest_group_periods)

-- linjegeometrier bakgrunnskart
drop table if exists traj;
create table traj as
select unit, st_makeline(geom order by time) as geom
from kadaver13_utm32_interp_15s
group by unit
```

Tillegg C.

Skjermbilder

C.1. Opprinnelig datasett

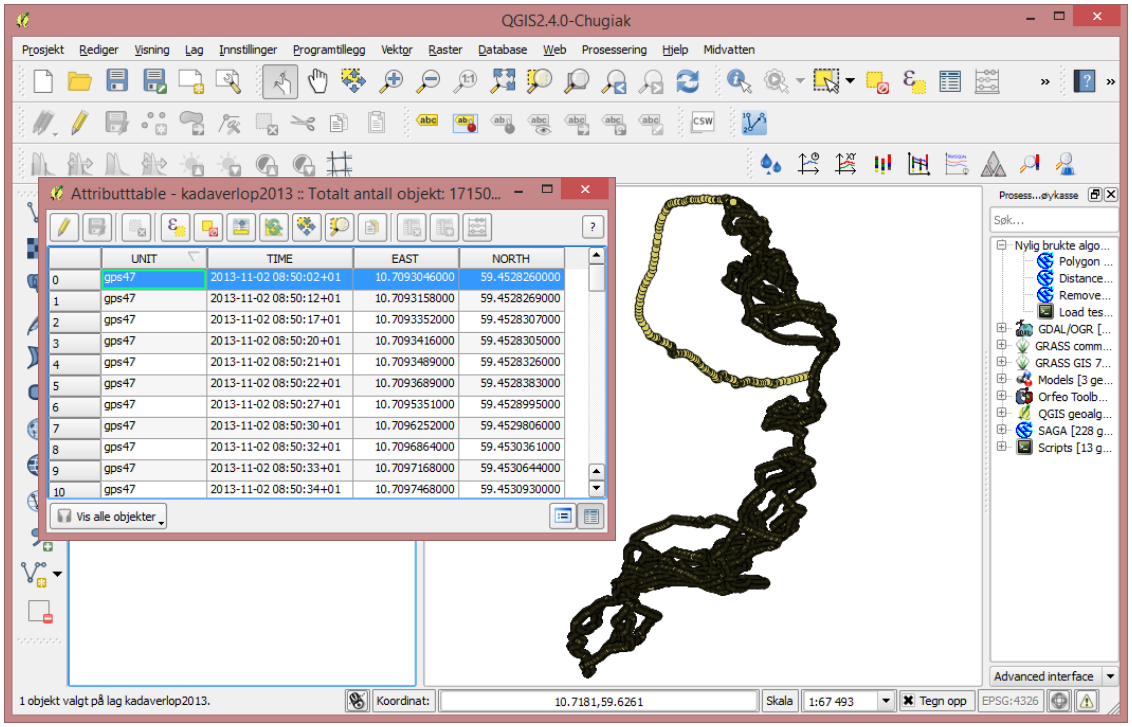


Fig. C.1.: Datasettet i shapefile-format

C.2. Oppdatert datasett

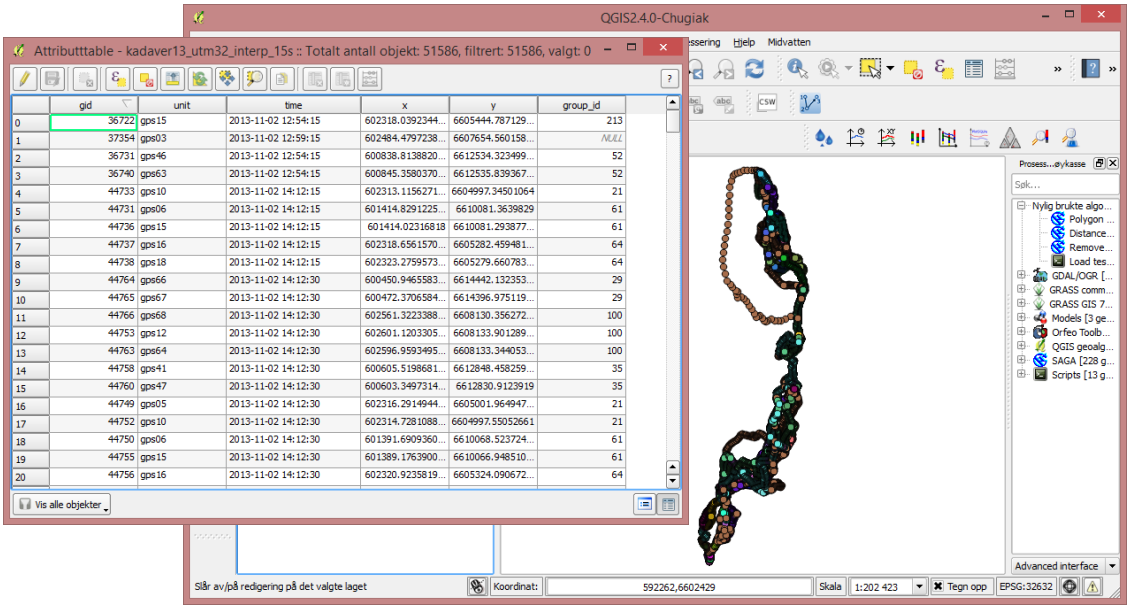


Fig. C.2.: Datasettet etter interpolering og iterativ clustering

C.3. Gruppetabell

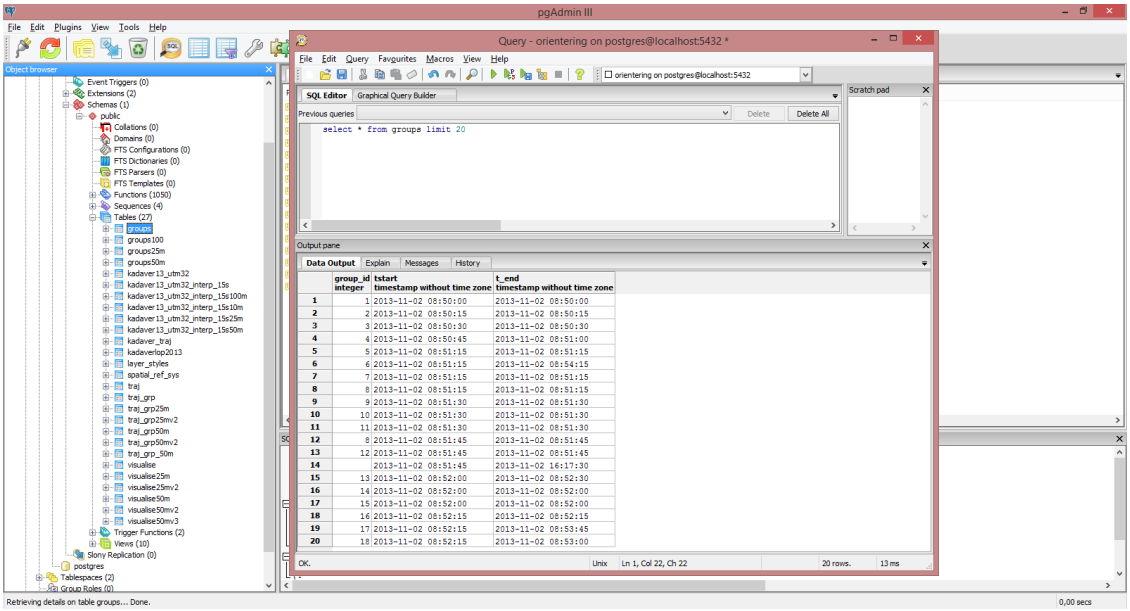


Fig. C.3.: Gruppetabell

C.4. Visualisering i QGIS

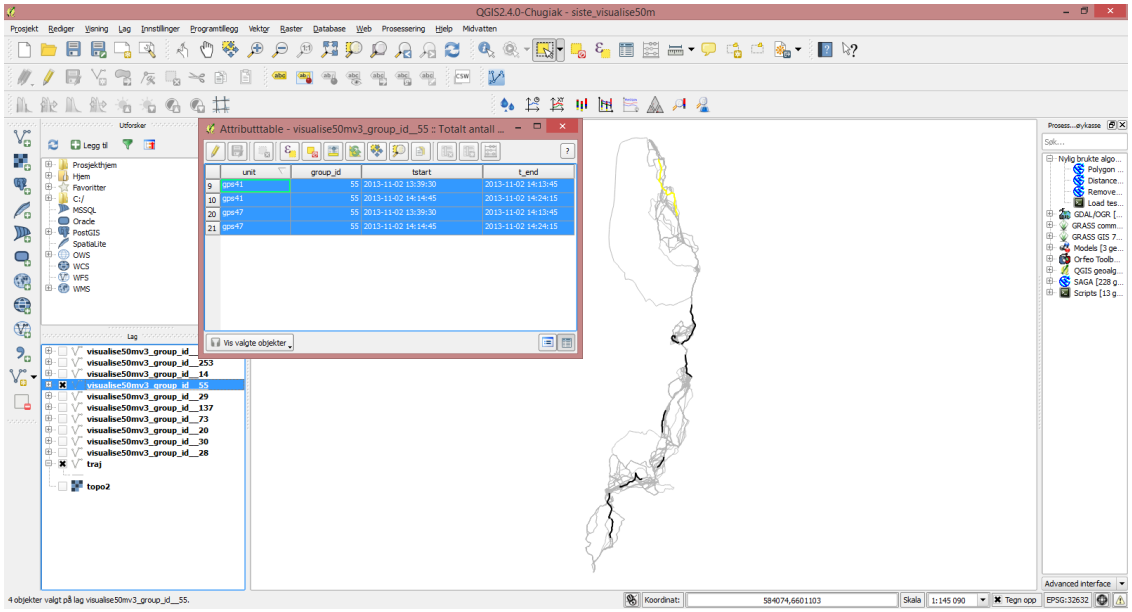
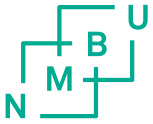


Fig. C.4.: Visualisering i QGIS



Norges miljø- og biovitenskapelig universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway