Blekinge tekniska
högskola – BTH

Examensarbete

GPS-baserad virtuell geografisk inhägnad för mobil enhet

Deltagare: Jim Gunnarsson, di98jgu@gmail.com

Handledare: Anders Nelsson, anders.nelsson@bth.se

Avdelning: Sektionen för datavetenskap och kommunikation

Källkod: https://github.com/di98jgu/GPS_ENCLOSURE

Datum:

Sammanfattning

Syftet med detta arbete är att finna en algoritm för att skapa och upprätthålla en virtuell inhägnad kring en mobil enhet. Området kan ha vilken form som helst och bör vara skalbar upp till och inkludera nationsgränser. Inhägnaden hanteras på den mobila enheten. Målsystemet är Android.

Mobiltelefoner, surfplattor och mer nyligen kameror är exempel på mobila enheter. En modern mobil enhet är en del av ett dator nätverk. Den är utformad för att vara uppkopplad hela tiden. En mobil enhet har förmågan att hantera stora mängder data; data som kan vara både känslig och ha ett ekonomiskt värde. Hantering av mobila enheter är avgörande för att skydda data och garantera säkerheten för hela nätverket.

En mobil enhet är en resurs i nätverket. Varje resurs definieras av en uppsättning parametrar. En parameter som gör en mobil enhet speciell är avsaknaden av fysiska gränser till världen. Typiskt så är det första steget i säkerhet åtkomstkontroll av resurser. Därför måste avsaknaden av fysisk åtkomstkontroll ersättas med andra medel, som att spåra den mobila enheten.

I detta arbete har jag undersökt möjligheten att skapa en virtuellt stängsel runt den mobila enheten. Den mobila enheten är fri att röra sig inom det område som begränsas av det virtuella stängslet. Den mobila enheten varnar alla berörda parter om den passera stängslet och därmed lämnar det tillåtna området. Alla nödvändiga åtgärder kan därefter vidtas.

Nyckelord: Android, Algoritm, Mobil enhet, Positionering, Position, larm

Summary

The aim of this paper is to find an algorithm to establish and maintain a virtual enclosure around a mobile unit. The area may take any form and should be scalable up to and including national borders. The enclosure is managed on the mobile unit. The target system is Android.

Smartphones, tablets and more recently, cameras is examples of mobile units. A modern mobile unit is part of a computer network. It is designed to be connected all the time. A mobile unit have the capability to handle large volumes of data, data that might be both sensitive and possess an economic value. Management of mobile units is vital to protect the data and insure security for the entire network.

A mobile unit is a resource in the network. Each resource is defined by a set of parameters. One parameter that make a mobile unit special is the lack of physical boundaries to world. Typically, the first step in security is access control to resources. Therefore, the lack of physical access control has to be replaced by other means such as tracking the location of the mobile unit.

In this paper I have studied the possibility of creating a virtual fence around the mobile unit. The mobile unit is free to move within the area enclosed by the virtual fence. The mobile unit alert all concerned parties if it crossed the fence and thus leave the allowed area. Any required action may then be taken.

Key words: Android, Algorithm, Mobile unit, Positioning, Position, alarm

Innehåll

1 Inledning

1.1 Mobil utveckling

Mitt val av examensarbete speglar mitt intresse för den tekniska mobila utvecklingen som telefon, surfplatta och andra mobila enheter. På 10 år har en mobiltelefon fullständigt transformerats från ett slutet system med en handfull funktioner jämte telefoni till ett system med nätverk, sensorer, och gränssnitt. Nya mobila enheter som surfplattan har tillkommit. Karakteristiskt för den mobila enheten är att ständigt vara uppkopplad och alltid tillgänglig. Kameran är nu på väg att omvandlas till en mobil enhet.

Utveckling sker kontinuerlig men är uppdelad faser. Datorn har gått från en maskin omgiven av en stab med operatörer till en burk på skrivbordet för att sedan krympa till en smidig laptop. Telefonin har gjort en liknande resa. När persondatorn kom så fick gemene man direkt tillgång till dator. Detta resulterade i en explosiv utveckling som ledde till helt nya användningsområden. Mobila enheter är ett sådant avgörande fasskifte, en språngbräda till något nytt.

1.2 Mobil säkerhet inom företag

Mobila enheter används allt mer i den dagliga verksamheten inom företag. De är numera inte bara ett kommunikationsmedel utan också ett viktigt verktyg av företagets verksamhet, en viktig del i trenden att kunna arbeta var som helst när som helst. Det innebär att känslig information är mer tillgänglig och hanteras på ett sätt som är betydligt svårare att kontrollera. Mobila enheter utgör en avsevärd säkerhetsrisk. I *Computer and Information Security Handbook* anges ett antal hinder för att upprätta god säkerhet[1]. De flesta av dessa kan direkt kopplas till mobila enheter:

- **Säkerhet är besvärligt från en användarens synvinkel:** Säkerhet kräver att resurser hanteras på ett bestämt sätt. Detta innebär ett merarbete för användaren vilket medför att resurshantering kan upplevas som både krångligt och tidsödande. En mobil enhet är en resurs som kräver ett visst handhavande.
- **Datorer är kraftfulla och komplexa:** En mobil enhet är en komplett dator med tillgång till Internet. Den innehåller och hanterar känslig information som kan ha ett ekonomiskt värde, typiskt företagshemligheter, nätverksinformation och persondata.

1 *Vacca, John R. (2009) Computer and Information Security Handbook [Elektronisk resurs]. Burlington: Elsevier - ISBN: 978-0-080921945*

- **Datoranvändare är osofistikerade:** De har liten eller ingen kännedom om säkerhet och de risker som finns. En mobil enhet är lätt att glömma eller av olika anledningar lämna obevakad.
- **Datateknik skapades utan en tanke på säkerhet:** Detta är inte sant vad gäller moderna operativsystem som Android. Android har ett stort antal inbyggda säkerhetsmekanismer. Ett exempel på detta är att vattentäta skott upprättas mellan applikationer och att tillgång till gemensamma resurser som kontaktuppgifter, Internet och GPS måste godkännas av användaren vid installation av applikationer.
- **Trenden är att dela information, inte att skydda den:** Ett typiskt exempel är sociala nätverk.
- **Information ska vara tillgängliga överallt:** Den mobila enheten finns med överallt och är ständigt uppkopplad. Det är viktigt att vara uppdaterad.
- **Säkerhet handlar inte om hårdvara och mjukvara:** Säkerhet handlar om att utveckla väl fungerande rutiner för hur hårdvara och mjukvara ska hanteras. Det är viktigt att användaren vet hur han eller hon ska agera i given situation.
- **Kriminella är mycket sofistikerade:** Immateriella tillgångar och kreditkortsuppgifter har ett stort ekonomiskt värde vilket har gett upphov till kriminalitet på internationell nivå. Med ett internt minne på 512 Mbyte och uppåt kan en typisk mobil enhet innehålla en ansenlig mängd information. En mobil enhet har dessutom som regel uttag för ett minneskort av önskad storlek.
- **Företag ser säkerhet som en utgift, inte som en investering.**

Det är därför viktigt att mobila enheter administreras på samma sätt som all annan datorutrustning. Ett viktigt verktyg för skydda den mobila enheten är att veta var den är. Positionering, som till exempel GPS mottagare, är de facto standard i de flesta enheter idag. Genom att upprätta en virtuell inhägnad runt enheten så kan man knyta den till ett bestämt geografiskt område. Man kan tänka sig flera olika områden där varje område har sina bestämda egenskaper. Den mobila enheten larma eller vidta andra åtgärder om den lämnar ett område. Ett tänkbart scenario är att den mobila enheten alltid ska befinna sig på företagets område.

2 Frågeställning

2.1 Definitioner

I det här arbetet används följande definitioner:

- En mobil enhet. En liten bärbar dator särskilt utformad för att vara i symbios mot Internet. Internet är helt centralt för en mobil enhet i sättet den är utformad och är tänkt att användas. Mycket av dess funktionalitet är beroende av Internet. Typiska exempel på mobila enheter är mobiltelefon, surfplatta och numera även kamera.
- Positionering är att bestämma den mobila enhetens geografiska position.
- Ett virtuellt staket är en sammanhängde uppsättning koordinater. Stängsel används omväxlande med staket.
- Ett geografiskt området har formen av en godtycklig månghörning, en polynom. Staketet löper runt områdets periferi.
- Att larma innebär att användaren av enheten informeras om att enheten har lämnat området.
- En virtuell inhägnad är ett geografiskt område omgärdat av ett virtuellt staket.

Dessa är definitioner är endast avsedda för det här arbetet. De sammanställs här för att undvika upprepning och underlätta läsandet.

2.2 Problemformulering

En given mobil enhet ska hägnas in med ett virtuellt staket med hjälp av positionering. Staketet omgärda och innesluter enheten så att den begränsas till ett bestämt geografiskt område. Ett larm ska aktiveras om enheten lämnar området.

Områdets storlek är inte strikt definierat. Sveriges landsgräns utgör en rimlig övre gräns. Det finns större geografiska områden som till exempel EU och USA men dessa är uppdelade i medlemsländer och stater. Minsta möjliga område begränsas av noggrannhet vid positionering.

De koordinater som utgör staketet hämtas från lämplig källa. Det finns stort antal möjliga källor men exakt vilken eller vilka är inte av vikt för detta arbete. Koordinaterna förutsätts utgöra hörnen i en polynom. Området är tvådimensionellt, ingen topografisk information används.

All hantering av inhägnaden sker på den mobila enheten.

2.3 Målsättning

Målsättningen är att bestämma de algoritmer som behövs för att upprätta och upprätthålla en virtuellt inhägnad. Att upprätta en virtuell inhägnad innebär att från en uppsättning koordinater skapa en virtuella inhägnad. Den virtuella inhägnaden upprätthålls genom att regelbundet kontrollera om den mobila enheten befinner sig innanför eller utanför staketet.

Detta är vad som skall göras:

- Att hitta en lämplig källa med koordinater för att skapa ett geografiskt område. Ett eller flera områden behövs för att utvärdera algoritmen.
- Att bestämma ett koordinatsystem för det virtuella området och hur koordinaterna ska översättas till detta.
- Att hitta den eller algoritmer som behövs för att bestämma om en godtycklig punkt är innanför eller utanför en given polynom.
- Att skriva en demo applikation som implementera algoritmen så att en virtuell inhägnad kan upprättas och upprätthållas på en mobil enhet. Demo applikationen skall ha följande funktionalitet:
  - Att upprätta en virtuell inhägnad från en uppsättning koordinater.
  - Att bestämma aktuell position.
  - Att bestämma om enheten är innanför eller utanför den virtuella inhägnaden.
  - Att logga och redovisa nödvändig information för att bedöma algoritmens lämplighet.
- Att bedöma om algoritmen är praktiskt användbar.

Arbetet kan utökas och anpassas beroende på hur mycket tid som finns tillgänglig och vilka problem som uppstår. Detta sker i så fall med algoritmen för inhägnaden i fokus. Utökning kan till exempel innebära:

- Att testa flera olika enheter för ökad förståelse av hur väl systemet fungerar i verkligheten.
- Att finna andra algoritmer som kan vara av intresse.
- Att undersöka avvikelse från sann positionen vid positionering och vad som kan göras för att kompensera för detta.

2.4 Förutsättningar

Avsikten med detta arbete är att finna en lämplig algoritm eller algoritmer för en mobil enhet och testa den eller de på en specifik enhet. Vi har följande:

- Operativsystem: Målsystem är Android vid utveckling av algoritmen. Valet av Android motiveras med att utvecklingsmiljön för Android är öppen och lättillgänglig. Det krävs inget avtal eller registrering för att utveckla för Android. Dokumentationen är omfattande och finns tillgänglig på Internet.
- System: Algoritmerna ska vara oberoende av en specifik mobil enhet, med detta avses både operativsystem, hårdvara och tjänster. En typisk tjänst är till exempel Google vilket skulle kräva att Google API är installerat på den aktuella enheten[2].
- Algoritm: Algoritmerna ska vara väl lämpade för rådande förutsättningar. En mobil enhet har till exempel begränsad batterikapacitet, minne och förmåga att positionera.
- Område: Algoritmen ska vara skalbar från ett ska kunna hantera ett så litet område som möjligt upp till Sveriges landsgräns.
- Dokumentation: Algoritmerna ska dokumenteras så att de lätt kan implementeras på andra typer av mobila enheter.

2.5 Avgränsningar

Följande avgränsningar för detta examensarbete är:

- Hårdvara: Att systematiskt testa olika mobila enheter. Detta kräver omfattande tillgång till hårdvara. Endast en eller ett fåtal enheter kommer att användas.
- Applikation: Att utveckla en för vanliga användare fungerande applikation. En applikation behöver utvecklas för att testa och utvärdera den virtuella inhägnaden. Den är endast avsedd för utvecklare. För vanliga användare fyller den ingen funktion.
- Applikation: Att låt användaren rita upp eller på annat sätt ange koordinater för en inhägnad. Detta skulle innebära att användaren tillåts modifiera eller skapa ett nytt område från ett gränssnitt av något slag. Detta tillför inget till målsättningen med detta arbete.
- Antal områden: Att hantera flera områden. Det antas här att om vi kan hantera ett område så kan flera hanteras. Med detta avses implementation och praktiska tester. Valet av algoritm sker med målsättningen att flera områden ska kunna hanteras.

2 *Google Developers (u.å.) Hello Map [Internet] Från: developers.google.com/maps [Hämtad: 1 okt 2013]*

3 Resultat

3.1 Mobil enhet

3.1.1 Android

Android är uppdelat på flera versioner. De två vanligaste versionerna är Gingerbread, Ice Cream och Sandwich Jelly Bean. Den äldsta av dessa är Gingerbread[3].

Vid utveckling av en Android applikation är det viktig att välja version. Android utvecklas mycket snabbt med nya funktioner. En mobil enhet med en äldre version kan inte använda en applikation utvecklad för nyare version av Android. Detta har ingen betydelse för att hantera den virtuella inhägnaden. Men det underlätta test av inhägnaden om demo applikationen stödjer en äldre version av Android. Fler enheter kan användas för att testa den virtuella inhägnaden.

3.1.2 Hårdvara

En mobil enhet karaktäriseras av att den har en processor med god kapacitet. Så pass god att java används som primärt programmeringsspråk i Android. Det var uppenbart en klokt val. Det räcker därför med att skatta stora O för att bedöma olika algoritmer.

Typiskt så har en mobil enhet 512 Mbyte i RAM minne och uppåt. Moderna operativsystem maximera användning av minne för att minimera svarstid. På Android så innebär det att applikationer försätts i dvala hellre än att helt stänga ner dem. För att dessa vilande applikationer inte ska ta upp allt minne så finns särskilda metoder för att återkräva minne. Administrativa metoder för att spara undan data kan därför behövas.

Batterikapacitet är den sista stora barriären vad gäller mobila enheter. Det är lätt att skapa applikationer som snabbt dränera all tillgänglig energi. Satellitpositionering tillsammans med radio tar mycket energi och bör därför användas så sparsamt som möjligt. Ett sätt att göra detta är att dynamiskt reglera tiden mellan en positionering till en annan.

Det är lämpligt att reducera omfattande beräkningsoperationer vid regelbundet återkommande uppgifter som att kontrollera att användaren befinner sig inom inhägnaden[4]. Detta är typiskt en bakgrundsuppgift och samsas med andra bakgrundsuppgifter.

---

3 *Android Developers (u.å.) Platform Versions [Internet] Från: developer.android.com/about/dashboards/index.html [Hämtad: 2 okt 2013]*

4 *Android Developers (u.å.) Performance Tips [Internet] Från: developer.android.com/training/articles/perf-tips.html [Hämtad: 3 Sep 2013]*

3.2 Positionering

3.2.1 Positionsleverantörer

I Android används flera olika tekniker för att tillhandahålla positionering[5]. Dessa benämns positionsleverantörer, eng. *location Providers*, vilka har sin unika uppsättning av styrkor och svagheter. De olika typerna av leverantörer komplettera varandra genom att vara lämpliga i olika situationer.

3.2.2 GPS och satellitbaserade system

Med GPS så bestäms aktuell position hjälp av satelliter i omloppsbana runt jorden. Det behövs minst tre för att bestämma longitud och latitud. Precisionen är hög men kräver fri sikt till satelliterna. Därför fungera tekniken dåligt inomhus. Signalen kan studsa mot byggnader och liknande vilket förvränger signalen och minskar möjligheten att bestämma position. Detta gör att GPS fungera mindre bra i urbana miljöer.

3.2.3 Trådlösa nätverk WI-FI

Trådlösa nätverk kan användas för att bestämma enhetens position. Enheten samlar information om de basstationer som finns inom räckhåll. Informationen skickas till Google som tillhandahåller känd positionsdata om basstationerna. Varje basstation har en unik MAC adress och kan därför identifieras. Det är ett ömsesidigt utbyte av information. Google samlar på detta sätt systematiskt in data om basstationer. En basstation kan dock lätt flyttas och därmed så stämmer dess position inte längre.

3.2.4 Mobiltelefonnätet

Varje mast har ett unikt id vilket gör möjligt att bestämma position på motsvarande sätt som för trådlösa basstationer. Då en mobil enhet i regel har täckning så finns goda möjligheter att bestämma enhetens position. För öka precisionen så kan information om föregående master användas.

3.2.5 Passiv positionsbestämning

Passiv positionsbestämning innebär att applikation lyssnar efter uppdateringar av positionen men utan att själv begära en uppdatering. Detta används för program som körs i bakgrunden. Poängen är applikationen alltid har tillgång till senast kända position, användaren behöver aldrig vänta på att applikation ska hämta in aktuell position.

5 *Milette, Greg & Stroud, Adam (2012) Professional Android Sensor Programming [Elektronisk resurs], John Wiley & Sons - ISBN: 978-1-118-18348-9*

3.2.6 Positionering i praktiken

Det finns lite information om hur väl positionering fungera i praktiken. Svårigheten ligger i att den information som finns endast beskriver mjukvaran, alltså Android, men inte hårdvaran. Det är rimligt att finns stora skillnader mellan olika mobila enheter.

Generellt så gäller att GPS positionering är långsammast, upp till flera minuter för erhålla aktuell position, men ger störst precision. Enligt lantmätriet ca 10 meter. Nätverksleveratörer, alltså trådlösa nätverk och mobilnätverk är snabbare mer ger sämre precision. De har fördelen mot GPS att de drar mindre energi.

3.3 Virtuell geografisk inhägnad

3.3.1 Definition av område

Ett geografiskt område beskrivs av en konkav eller konvex polygon. Polygonen byggs upp av ett antal punkter, koordinater. Att polygon beskriver ett geografiskt område innebär att polygonen sluten och väldefinierad. Det sista innebär att överlappande linjer och eller koordinater inte får förekomma. Systemet ska kunna hantera ett större antal koordinater. Sveriges landgräns inkluderat kust är cirka 5400 km. Så en övre gräns på 5000 koordinater bör vara fullt tillräckligt.

Enheten antas befinnas sig i området även om inget hindra att samma algoritm används för att exkludera enheten från ett givet område. Enheten är en till antalet, något som inte alls är självklart ty man kan lätt föreställa sig en situation med flera enheter där deras inbördes relation till varandra är av intresse. Men detta är en helt annan frågeställning.

Bristande noggrannhet vid positionering gör det svårt att strikt bestämma om enheten är ute eller inne. En lösning är att upprätta två områden. Ett inre område för att varna och ett yttre för att larma. Detta förutsätter att skillnaden mellan bestämd position och faktisk position ej överstiger ett visst värde.

3.3.2 Koordinatsystem

Ett staket kan antingen skapas av användaren på den aktuella enheten eller tillhandahållas från extern källa. Både är rimliga. I det här fallet så är enheten en klient i ett nätverk som administreras centralt.

Skapas staketet på själva enheten så kan man tänka sig att området ritas eller annat sätt markeras. Google maps som finns tillgängligt på Android och utgör, åtminstone på Android, enkel lösning. Apple tillhandahåller numera egna kartor för Iphone.

En annan lösning är att hämta gränser från befintligt kartmaterial. Kartmaterial från Lantmäteriet anger koordinater enligt RT90 eller SWEREF 99. Gränsen går vid 2007 då Lantmäteriet gick från RT90 till SWEREF 99. Detta är plana koordinater och måste översättas till GPS koordinater. Detta förutsätter att enheten kommer att användas i Sverige.

Det finns inget behov av att mäta faktiska avstånd eller vinklar. Detta underlätta då översättning från ett koordinat system till ett annat är en komplicerad procedur. Inhägnaden kommer internt att arbeta med enhetslösa x och y koordinater. All översättning mellan olika koordinatsystem ligger utanför hanteringen av inhägnaden. Detta innebär att översättning av olika koordinat system är en problematik skild från hantering av inhägnaden.

3.3.3 Egenskaper hos enhet

Området antas vara stort i förhållande till rörelsehastigheten hos enheten. Det betyder att det tar en viss tid för användaren av enheten att flytta sig från en sidan av inhägnaden till den andra. Detta är inget krav men spar batteri ty positionering kan ske med större tidsintervall.

Användarens rörelsemönster är inte slumpartat utan beror av vad användaren befann sig vid föregående tillfälle. Vidare om enheten bärs av en användare till fots så kan vi anta att enheten rör sig lite eller inte alls under längre perioder.

Det är alltså lönt att skatta rörelsehastighet hos enheten. Det gör att mindre avbrott vid positionering kan kompenseras. Enheten kan avvakta med att rapportera saknad position och göra ytligare försök att fastsälla enhetens position. Rörelsehastighet är också användbart för att bestämma behov av positionering.

Enklast är att skriva in enheten i ett fyrkantigt delområde innanför inhägnaden. Detta mindre område har kända dimensioner och med skattad rörelsehastighet och senaste position vet vi när användaren tidigast kommer lämna delområdet. När enheten lämnar delområdet upprättas ett nytt delområde runt enheten.

3.3.4 Flera områden

Det finns ett intresse av att hantera flera områden. Behovet av resurser ökar i direkt proportion till antalet områden. Skrivs enheten in ett delområde blir hantering av flera områden enklare. Vi vet direkt i vilket eller vilka områden som enheten befinner sig. Man kan tänka sig att skapa relationer mellan de olika områdena.

Bortser man från relationer så är flera områden endast en lista med enskilda områden. Att hantera flera områden blir då en mer administrativ till sin karaktär. Relationer mellan områden underlättas om delområdena är geometriskt enkla som cirklar, trianglar och rektanglar.

3.4 Algoritm för virtuell inhägnad

3.4.1 Val av algoritm

Målet med de två föregående kapitlen var att bestämma algoritmens egenskaper algoritmen. Genom att skatta användarens rörelse mönster så kan systemet göras mer tillförlitligt. Det ska gå snabbt att bestämma om användaren befinner sig i visst område eller inte. Ett delområde

upprättas runt användaren för att minimera behovet av att uppdatera GPS positionen. Vi kan spara undan data så att en initialt tidskrävande algoritm bara behöver köras en gång. Det krävs en ökad administration men att skyffla data är inget problem.

Det finns ett rikt utbud på olika algoritmer för polygoner. Den enklaste av dessa är låta en tänkt linje utgå från enheten till yttre gränsen för polygonen. Genom att räkna antalet gånger som linjen skär polygonen vet vi om vi inne eller ute. Udda antal skärningar så är enheten inne och jämt antal ute. De yttre gränserna kan enkelt bestäms genom att gå ett varv runt polygonen.

Den här typen algoritm är enkel och kräver inget minne. Den har två problem. Vi behöver kortaste avstånd till kant från given punkt i polygonen. Detta kräver att vi räknar på varje delsträcka. Det andra problemet är att algoritmen kräver ett varv runt polygon. Det ger komplexiteten stora O(n) där n är antalet koordinater. Det helt acceptabelt för ett mindre antal koordinater men vi har som målsättning att kunna hantera mer än 1000 koordinater.

Den mest lämpade och intressanta som jag funnit är att dela in polygonen i rutor. Grundidén här att rutorna delas in i tre kategorier. De är inne, ute eller innehåller en del av polygonen. Detta gör det går snabbt att bestämma om enheten är inne eller ute ty även då enheten befinner sig inom en ruta som korsas av polygonen så är beräkningarna reducerade till endast den rutan.

I bästa fall så är komplexiteten O(1) i värsta fall går den mot O(n). Det första fallet får vi om enheten antingen befinner sig ute eller inne. Det senare är specialfall, merparten av alla koordinater är samlade på ett mycket litet område. Detta är normalt inte trolig utan vi kan anta koordinaterna är jämt fördelade.

Låt oss anta att vi har en rektangel liknande polygon på 1000 koordinater och rutnätet är 30 gånger 30 rutor. Omkretsen är då 118 rutor eller 13 % av alla rutor upptas av polygonen resten är innanför och inga utanför. Rutnätet är optimerat för den aktuella polygon och är endast något större än polygonens min och max värde. Antalet koordinater per ruta är mindre än 10.

Nackdelen med ett rutnät är att det initialt krävs en hel del räknade. Varje delsträcka som ingår i polygonen behöver räkas ruta för ruta. Vi får O(n*m) där n är antalet koordinater och m är antalet rutor som en viss delsträcka passera. En diagonal från hörn till hörn skär maximalt antal rutor men vi kan sätta m till sidan av rutnätet då konstanten faller bort, alltså 30 i fallet ovan. Men även detta är ett specialfall, i verkligheten blir få eller inga sträckor så långa.

Det tillkommer ytligare initialt arbete. Vi måste säkerställa att inga koordinater eller sträckor överlappa. Det löser delvis kravet på en väl definierad polynom. Två sträckor kan fortfarande ligga mycket nära varandra.

3.4.2 Upprättande av inhägnad

De två viktigaste delarna för att upprätta ett rutnät är dela upp polynom ruta för ruta och att bestämma om rutornas hörn är inne eller ute. En given polynom är en serie av av segment där varje segment behandlas var för sig. Studera vi ett segment ab från en given polygonen så har vi tre grundläggande fall, se bild 1.

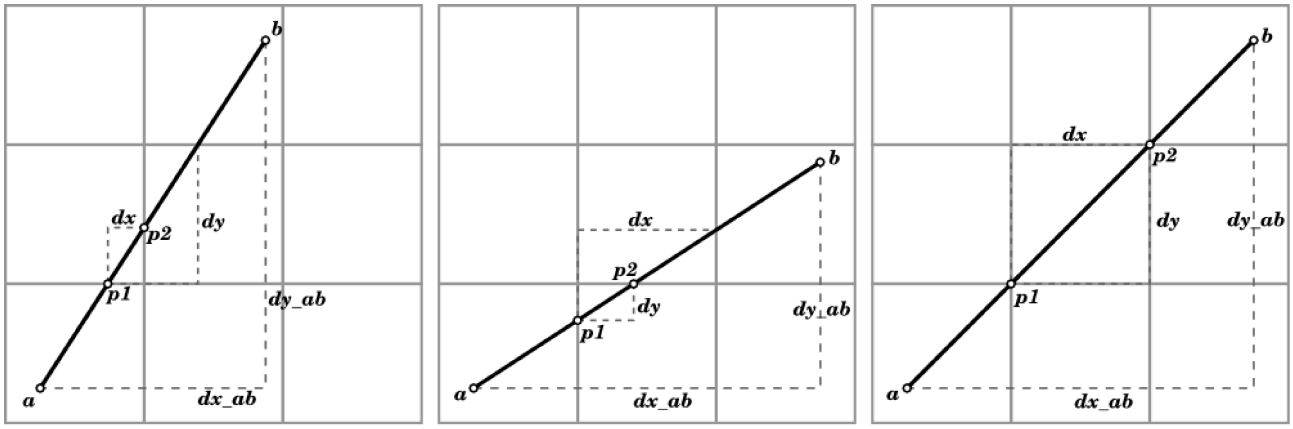

Bild 1: Ett segment från en given polygon

Betrakta segmentet som hypotenusan i rätvinklig triangel. Vi stycka upp segmentet i delsegment där segmentet skär rutnätet. Vi får då en delsträcka a till p1 och p1 till p2 och så vidare. Summan är av de olika sträckorna är den totala längden på segmentet.

Anta att vi står i punkt p1 och söker p2. Avstånd till rutnät i x och y led kan lätt bestämmas ty rutnätets dimensioner är kända. Detta ger dy och dx som är kateter i två trianglar av olika storlek. Den minsta av dessa, med kortaste hypotenusa, ger oss p2. För senare beräkningar är det också viktigt att spara information om var segmentet skär cellväggarna, sida tak eller golv.

Ett special fall kan inträffa när längden på hypotenusan är lika, bild 1 till höger. Detta är ett typiskt exempel på ett gränsvärde om inte utgör något realt problem men som lätt missas vid implementation. Lösningen är att konsekvent låta segmentet skära sidan eller taket på den aktuella cellen.

När samtliga segment har skrivits in i rutnätet så polygonen uppdelad cell för cell. Tre typer av celler finns, inre yttre och inhägnad. Djupare information behövs endast om de celler som innehåller en del av inhägnaden. Det lönar sig att använda dubbel bokföring, en lista med typ av cell och en med inhägnad.

Att plocka fram en cell för given position kan nu göras men för att bestämma om positionen är inne eller ute så behövs en referens. Hörnen i cellen är väl definierade, se bild 2. Hela den yttre ramen med alla dess hörn är ute. Vid a så skär polygon linje 1 i y led så hörnet 11 är inne. Hörnen växlar mellan att vara inne och ute vid b. Hela rutnätet avläses linje för linje.

En cell kan ha flera oberoende linjer som i c. Det betyder att polygonen här skär y linjen två gånger och hörn 43 är inne. En paritets bit eller motsvarande behövs alltså för varje cell för räkna antal skärningar.

Med hörnen som referens så är det möjligt att bestämma om en given position är inne eller ute. En rätt linje dras från den aktuella positionen mot önskat hörn. Antal gånger som linjen skär polygonen samt om hörnet är inne eller ute bestämmer om den givna positionen är innanför eller utanför inhägnaden. Studera bild 2 för att bekräfta detta.

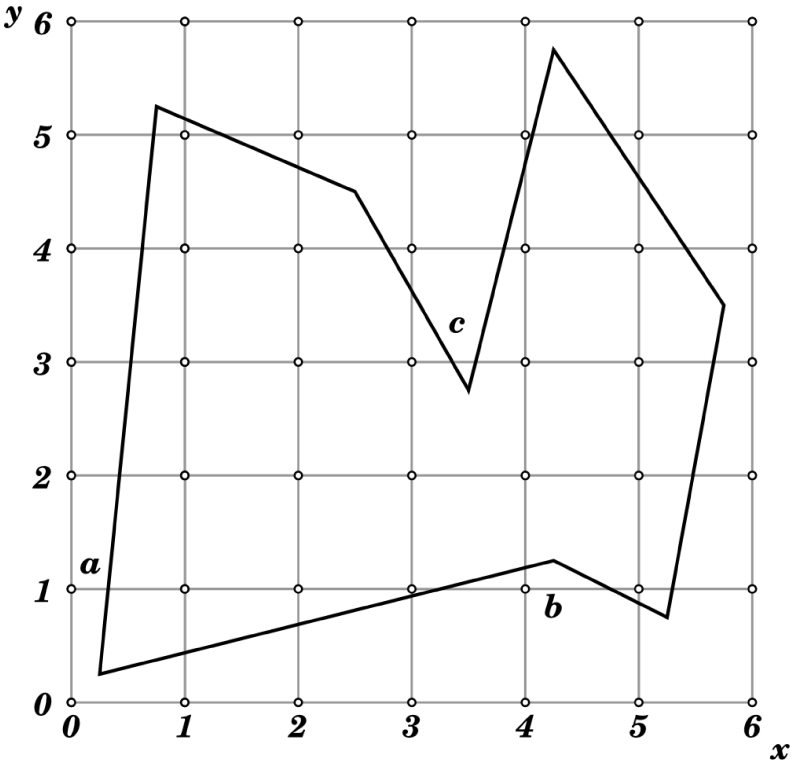

Bild 2: Polygonen inskriven i rutnätet.

Rutnätets dimensioner bestäms av inhägnaden och önskad upplösning. Min och max värde för inhägnaden bestäms först och från detta kan önskade dimensioner för rutnätet beräknas. Upplösning kan vara men behöver inte vara antal rutor i x och y led. Har vi till exempel flera områden så hängs de upp i ett gemensamt rutnät. En annan definition av upplösning kan då vara lämplig.

Att skriva in inhägnaden i rutnätet består alltså av följande delar:

- Beräkna rutnätets dimensioner.
- Att skriva in inhägnaden ruta för ruta.
- Att bestämma om rutnätets hörn är inne eller ut.

3.4.3 Odefinierat område

Inhägnaden kan inte upprättas om det finns odefinierade områden orsakade av korsande linjer. Att bestämma om två linjer korsar varandra kan göras genom att kontrollera varje segment mot alla andra segment. Detta är en uppenbart ineffektiv metod vars tidsåtgång är kvadratisk till antalet koordinater. Det finns mer effektiva lösningar som ger O(n log(n)) men dessa för den här uppgiften orimligt komplicerade. Sänks kravet och lokalt odefinierade områden inom en ruta accepteras så kan kontroll göras mycket snabbt, i storleksordningen O(n) där n är antal rutor med inhägnad.

Med ett lokalt odefinierat område menas att ett eller flera delpolynom överlappar varandra så att ett slutet område uppstår inom en ruta, se bild 3. Lokalt odefinierade områden kan accepteras då dessa inte påverka förutsättningarna för upprättandet av inhägnaden och användandet av denna i sin helhet. Korsar två delpolynom varandra så faller grunden för rutnätet då in och utsida ej kan fastställas.

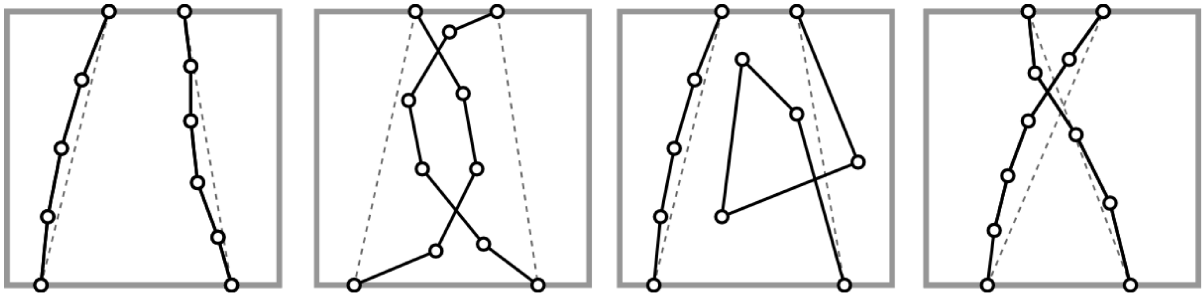

Bild 3: Lokalt odefinierade områden och korsande linjer

Kontroll görs genom att varje delpolynom ersätts med en rät linje från ingång till utgång, streckad linje i bild 3. Korsar dessa linjer varandra så har vi en ogiltigt inhägnad. Problemet med lokalt odefinierade områden reduceras med ett allt mer finmaskigt rutnät. En lämplig tumregel kan vara att att det totala antalet rutor ska vara av samma storleksordning som antalet koordinater.

3.5 Implementation på Android

3.5.1 Demo applikation

Avsikten med applikationen är att testa den virtuella inhägnaden på en faktisk mobil enhet. Applikationen är inte avsedd för vanliga användare. De funktioner som behövs:

- Att starta och stänga ned applikationen
- Att välja område.
- Att aktivera den virtuella inhägnaden.
- Att informera om enheten lämnar inhägnaden.
- Att visa information som kan vara av intresse för utveckling och felsökning.

Ett antal områden är bestämda på förhand och kan varken skapas eller på något sätt modifieras. De har skiljer sig åt genom områdets storlek, position och antal koordinater. Syftet med de olika områdena är dels att testa algoritmen och dels att felsöka koden.

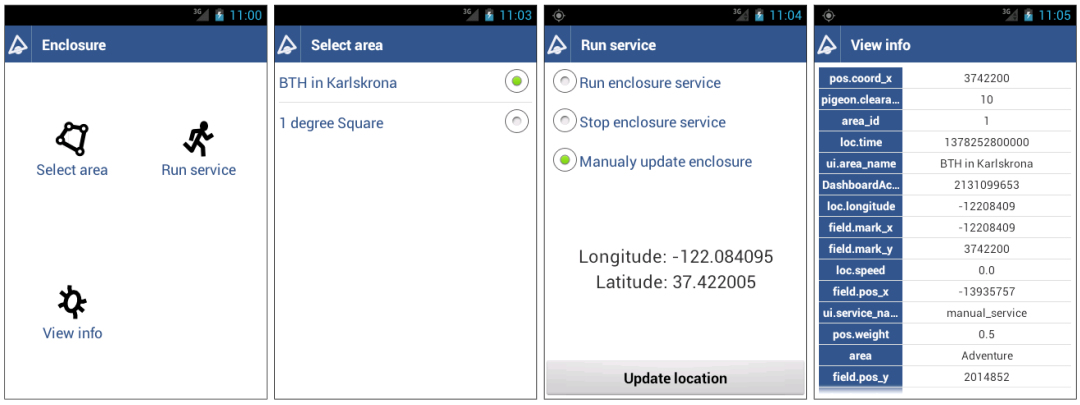

Bild 4: Demo app i Android.

Användaren informeras med ett textmeddelande om enheten lämnar den virtuella inhägnaden. Ett nytt meddelande visas om enheten återvänder till inhägnaden. En enhet kan vara innanför, utanför och vid inhägnaden. Varje gång enheten växlar tillstånd så visas ett meddelande.

Information relevant för utveckling av koden är olika interna tillstånd, position och status för inhägnaden.

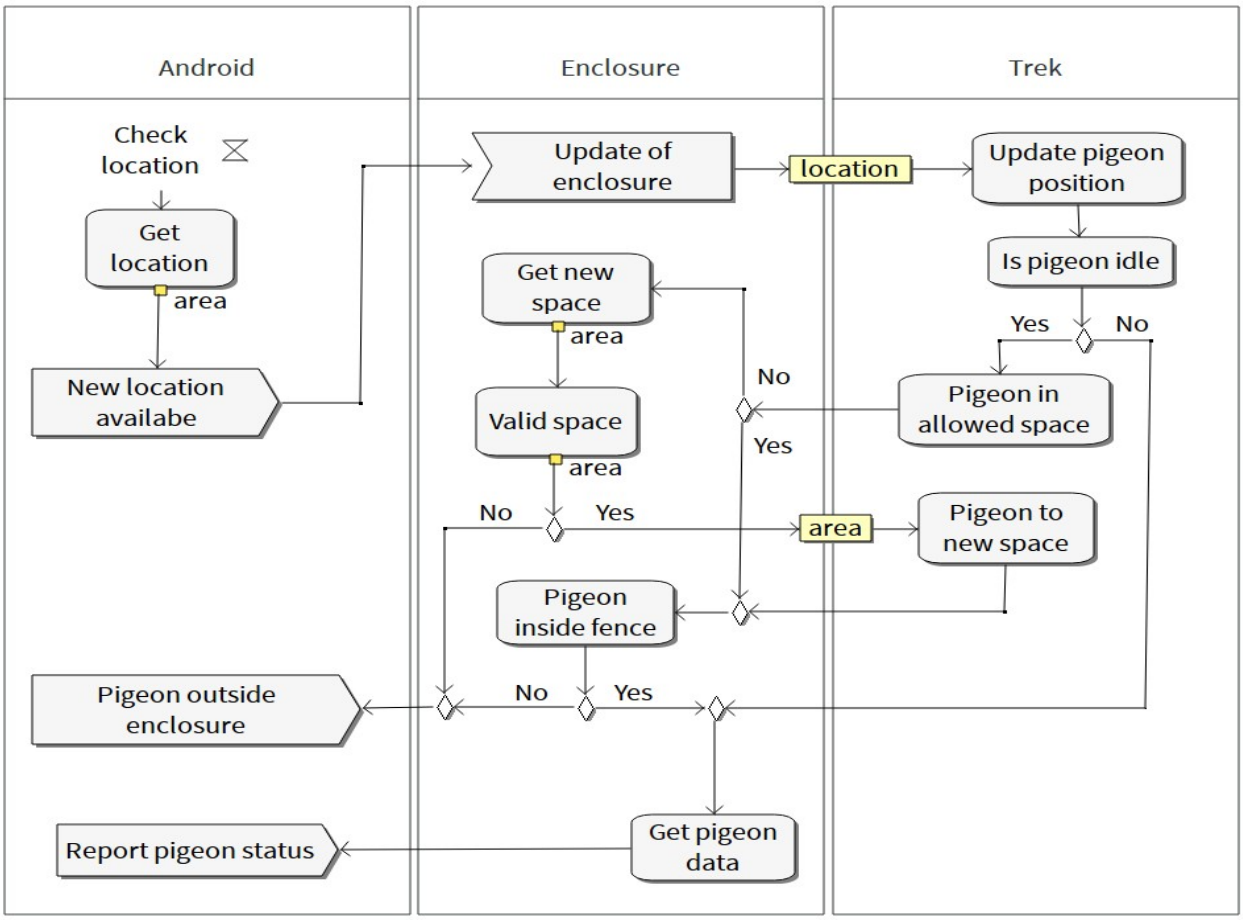

Bild 5: Aktivitetsdiagram över uppdatering av position och inhägnad.

3.5.2 Uppdatering av position

Detta är en service som aktiveras med bestämda tidsintervall. Bild 4 visar ett övergripande aktivitetsdiagram. En service i Android är ett byggnadsblock för hantera bakgrundsuppgifter[6]. Aktivitetsdiagramet visar de viktigaste delarna, `Android`, `Trek` och `Enclosure`. Inhägnaden hanteras av `Enclosure` medan `Trek` hantera enhetens nuvarande position och var enheten är i förhållande till ett givet delområde. För att bestämma om enheten är innanför inhägnaden behöver följande uppgifter utföras:

- Hämta in senaste position, var den mobila enhet befinner sig just nu. När en ny position finns tillgänglig så skickas den information som en händelse.
- Uppdatera den mobila enhetens position och kontrollera om enheten förflyttat sig. Har enheten inte rört sig sedan senaste position så finns ingen behov av vidare kontroll. Enheten antas vara innanför inhägnaden.
- Kontrollera om enheten befinner sig inom ett givet delområde. Ett delområde är en mindre del av inhägnaden. Det kan innehålla en del staketet.
- Skapa ett nytt delområde. Detta sker endast om enheten har lämnat föregående delområde. Går det inte att skapa ett nytt område så är enheten utanför staketet.
- Kontrollera om enheten befinner sig innanför staketet. Denna sker efter det att delområdet har uppdaterats. Är enheten utanför staketet så skickas en händelse.
- Sammanställa information om den enhetens aktivitet. Denna information används för att bestämma behov av positionering.
- Skicka information om enhetens senaste aktivitet.

Att bestämma aktuell position görs med hjälp `LocationManager`. Här har jag utgått från Reto Meiers metod för positionsbestämning[7]. Det som särskiljer hans hans metod för att bestämma aktuell position är han först gör en passiv positionering. Det spara energi om det finns andra applikationer som också använder positionering. Om senast kända position inte uppfyller vissa kriterier så begärs en ny positionering.

Med en ny position så kan enhetens position uppdateras vilket sker i `Trek`. Enheten representeras av klassen `Pigeon`, ett namn valt i bristen på ett bättre. Klassen `Pigeon` har en position, rör sig med viss hastig och befinner sig i ett bestämt område. Lämnar `Pigeon` sitt utstakade område så informera `Pigeon` om detta till eventuella lyssnare.

Befinner sig enheten, `Pigeon`, på eller i närheten av en viss position under en längre tid så anses användaren vara inaktiv. Någon kontroll om enheten befinner sig inom sitt område behöver bara göras om användaren är aktiv. Detta ger systemet en justerbar grad av tröghet vid positionering. Det område som `Pigeon` befinner sig inom är ett delområde av hela inhägnaden. Ett nytt delområde skapas vid behov av `Enclosure`.

---

6 *Android Developers (u.å.) Services [Internet] Från: developer.android.com/guide/components/services.html [Hämtad: 4 Sep 2013]*

7 *Meier Reto (23 juni 2011) Android Developers Blog, A Deep Dive Into Location [Blog] Från: android-developers.blogspot.se/2011/06/deep-dive-into-location.html [Hämtad: 4 Sep 2013]*

För att bestämma tidpunkt för nästa positionering så behövs enhetens rörelsehastighet över en viss sträcka. Denna information tillhandahålls av `Pigeon`. Den tid det tar med nuvarande rörelsehastighet att nå gränsen för utstakat område är maximalt tidsintervall för nästa positionering. Att schema lägga tidsbestämda händelser görs med `AlarmManager` och beskrivs av Marko Gargenta i hans *Android Bootcamp Serie*[8].

3.5.3 Positionering och koordinater

Vid implementation används råa koordinater, det vill säga grader. Ingen omvandling sker från grader till ett plant koordinatsystem. Detta påverka inte inhägnaden och dess funktion. Koordinater för att faktiskt område som till exempel BTH har hämtats från hitta.se[9]. Samtliga områden är samlade i en statisk klass `Route`. Detta gör att nya områden lätt kan läggas till vid behov.

3.5.4 Inhägnad

Inhägnaden upprättas och upprätthålls av `Enclosure`. För att upprätta det rutnät som utgör grunden i algoritmen så behövs ett plan. Detta plan, basklassen `EuclidPlane`, är grunden i inhägnadens koordinatsystem. Andra klasser som använder koordinatsystem är `Boundary` och `Segment`. Klassen `Segment` är central för att bestämma om en given koordinat är innanför eller utanför inhägnaden.

Upprättande av själva inhägnaden sker i klassen `Enclosure` och det är denna klass som också administrera inhägnaden. Dess huvudfunktion är att skapa de delområde som används av `Pigeon`. Algoritmen för att upprätta inhägnaden som beskrivits i kapitel 3.4 är implementerad i `Enclosure`.

Två tabeller används för spara information om inhägnaden. En tabell `TileTable` används för att spara data om de celler som innehåller en eller flera delar av inhägnaden. En cell som är tom sparas inte. Varje del av inhägnaden sparas i klassen `SectionTable`. Tre klasser behövs för att spara inhägnaden, `Grid`, `TileTable` och `SectionTable`. Det tillkommer också två gränssnitt `Tile` och `Section` som används för att administrera tabellerna.

8 *Gargenta, Marko [NewCircle] (7 mars 2012) 25 - Alarms and System Services: Android Bootcamp Series 2012 [Föreläsning] Från: www.youtube.com/watch?v=QEMk4SwsJMg&list=SPE08A97D36D5A255F [Hämtad: 4 Sep 2013]*

9 *hitta.se (u.å.) Karta, vägkarta, satellitbilder och gatubilder [Karta] Från: www.hitta.se/karta [Hämtad: 4 Sep 2013]*

3.6 Utvärdering

3.6.1 Val av algoritm

Vid implementation så framkom det att den valda algoritmen var betydligt mer komplex än väntat. Data för inhägnaden delas upp på enskilda celler. Rutnätet utgörs av ett antal celler. Varje cell kan ha en eller flera sektioner av inhägnaden och varje sektion är uppbyggd av ett antal koordinater. Den här strukturen gör det möjligt att hantera ett mycket stort antal koordinater men är svår att motivera för ett litet antal koordinater.

Det råder osäkerhet om hur stort området måste vara för att dra nytta av den rutnätsbaserade algoritm som här har användes för upprätta och upprätthålla inhägnaden. Detta beror på att det under arbetets gång inte har varit möjligt att hitta information om den mobila enhetens beteende. Två viktiga faktorer är hur fort den mobila enheten kan uppdatera sin position och med vilken precision en position kan bestämmas. Dessa faktorer bestämmer enhetens rörelsemönster inom inhägnaden.

Frågan som har uppstått är om den valda algoritmens verkligen löser uppgiften. Detta ändra förutsättningarna för utvärdering av algoritmen. Avsikten var att testa på algoritmen på en eller flera enheter. Men om algoritmen inte hantera uppgiften på önskat sätt så är det bättre att utvärdera projektet i sitt nuvarande tillstånd.

3.6.2 Vad är implementerat

All primär kod är implementerad. Detta innebär att applikationen kan starta och att all funktionalitet är implementerad. Inhägnaden upprättas och upprätthålls. Applikationen är funktionell men inte användbar. Den är allt för instabil för att installeras på en mobil enhet.

Fokus har legat på de två delarna `Enlosure` och `Android`, se bild 5. Att fokus kom att delas mellan `Enclosure` och `Android` beror på att undertecknad har begränsad erfarenhet av ekosystemet Android och hade inte tidigare arbetat med `LocationManager`. Den tredje delen `Trek` som hantera enhetens rörelse och position har implementerats så enkelt som möjligt.

Det som saknas är undantagshantering, *exceptions*, och systematisk testning med JUnit[10]. Löpande under utveckling har valda delar av koden testats i Python. Python är användbart för validering av algoritmer och beräkningar. Det kan också finnas andra problem som är mindre uppenbara som minnes läckage.

En förändring av den befintliga algoritmen skulle medföra begränsade förändringar av applikationen. Applikationen är uppdelad i avgränsade delar vilket underlätta även omfattande förändringar i algoritmen.

3.6.3 Storlek på geografiskt område

Den valda algoritmen utgår från att mobila enhetens hastighet är låg i förhållande till storleken på inhägnaden. Det är inte den faktiska hastigheten som är intressant utan hur mycket den mobila enheten har förflyttat sig inom inhägnaden från en positionering till en annan. Tid saknar

10 *JUnit (u.å.) [Hemsida] Från: junit.org [Hämtad: 5 Sep 2013]*

betydelse, det är avståndet mellan två positioneringar i förhållande till storleken på området som är det viktiga. Idealt så bör mobila enheten upprätthålla sig så länge som möjligt inom en cell då detta minimera den administrativa kostnaden att upprätthålla inhägnaden.

Avståndet mellan en positionering till en annan beror av användarens hastighet och hur fort den mobila enheten kan uppdatera sin position. Demo applikationen är implementerad med antagandet att den mobila enheten rör sig inom en cell och från cell till cell. Frågan är om detta är ett korrekt och rimligt antagande. Låt oss anta användaren rör sig med tre hastigheter, till fots 1 m/s, på cykel 5 m/s och med bil 25 m/s.

För en service, som körs i bakgrunden, så rekommenderas ett minsta tidsintervall på 5 minuter från en positionering till nästa[11]. Typiskt är detta en applikation kopplad till geografisk information eller sociala medier. Längre tidsintervall ger Android möjlighet att samordna de tjänster som använder radio. Radion är den del av systemet som tär mest på batteriet. På 5 minuter så hinner en person till fots ca 300 meter, på cykel 1,5 km och i bil 7,5 km. Detta ger ett stort område om den mobila enheten ska röra sig från cell till cell. Varje cell i rutnätet behöver vara en mil i fyrkant eller mer om användaren kör bil.

Det finns applikationer för bilnavigering. I detta scenario har den mobila enheten extern strömförsörjning från bilen. Aktuell position bestäms med stöd av karta och död räkning från senast kända position. Den mobila enheten uppdatera sin position löpande med minsta möjliga tidsintervall. Vi kan anta att radion är på hela tiden och att samtliga tillgängliga tekniker för positionering är aktiverade.

För bilnavigering ska fungera väl så behöver tidsintervallet vara i storleksordningen 10 sekunder eller bättre. En bil avverka 250 meter på 10 sekunder. Till fots så blir sträckan 10 meter och på cykel 50 meter. Kan en mobil enhet uppdatera sin position varje sekund så blir upplösningen enstaka meter till några 10-tal meter. Enligt lantmäteriet så är osäkerheten vid ren satellit navigering ca 10 meter[12]. En mobil enhet har tillgång till andra positionsleverantörer men dessa ger större osäkerhet. Det är möjligt att om samtliga positionsleverantörer används så kan osäkerheten minskas men detta är ingen funktionalitet som tillhandahålls av Android.

För en applikation där positionering är av central betydelse som i det här fallet så är det rimligt att offra batterikapacitet till förmån för tätare uppdatering av position. Med en positionering varje minut så är blir avstånden 60 meter till fots, 300 meter på cykel och 1500 meter med bil. Detta ger ett minsta område i storleksordningen 10 km. Området kan vara mindre men själva poängen med algoritmen är att området delas upp med ett rutnät.

Det är beteendet vid positionering som bestämmer hur fort den mobila enheten kan uppdatera sin position. Tätare positionering medför en ökad energi kostnad. Varje position är behäftad med en viss osäkerhet. Ett stort antal möjliga felkällor påverka graden av osäkerhet. Det går inte att bestämma någon nedre gräns för hur litet området kan vara för algoritmen ska fungera som det är tänkt utan kännedom om enhetens beteende. Det är enhetens beteende som sätter gränsen för vad som är möjligt vid positionering.

---

11 Android Developers (u.å.) LocationManager [Internet] Från: developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(long, float, android.location.Criteria, android.app.PendingIntent) [Hämtad: 5 Sep 2013]

12 Lantmäteriet (u.å.) WGS 84 [Internet] Från: www.lantmateriet.se/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/Referenssystem/Tredimensionella-system/WGS-84 [Hämtad: 5 Sep 2013]

3.6.4 Behov av positionering

Den grundläggande principen för den rutnätsbaserade algoritmen är att dela upp inhägnaden i enskilda celler där varje cell är oberoende av andra celler. En cell är ett delområde av den totala inhägnaden. Den mobila enheten rör sig inom ett delområde och från delområde till delområde. Det är så demo applikationen är implementerad. Detta innebär att det finns en stark koppling mellan delområde och rutnät. Rutnätet bestämmer storleken på delområdet. Storleken på delområdet bestämmer i sin tur hur ofta den mobila enheten behöver uppdatera sin position.

För minimera behovet av att uppdatera enhetens position så behöver delområdet vara så stort som möjligt. Det kan göras genom att söka av de celler som gränsar till den cell som ockuperas av den mobila enheten. Man kan tänka sig att avsökningen sker i ett spiralformat mönster. När en cell med en del av inhägnaden påträffas så kan delområdet upprättas. En annan betydligt enklare metod för att bestämma närmaste cell som innehåller en del av inhägnaden är att söka av rad för rad. Tidsåtgången är linjär för bägge metoderna. En spiralformad avsökning har som bäst O(1) och som värst O(n) där n är antalet celler.

Ett annat sätt att minska behovet av positionering är att låta delområdet växa för varje ny cell som den mobila enheten ockupera. Men detta ger ett oregelbundet område vilket gör det svårare att skatta tiden det tar för enheten att lämna området om den rör sig med oförändrad hastighet. Hastighet kan kompletteras med rörelseriktning vilket gör denna metod mer rimlig. Att administrera ett dynamiskt växande delområde kräver dock betydande merarbete.

En cell är en självständig enhet som innehåller all information om det geografiska område den representera. Det möjligt att för varje cell bestämma minsta avstånd till inhägnad. Man kan till exempel utgå från centrum av cellen. Detta mått kan användas för att skapa ett delområde. Fördelen med denna metod är att den ligger i linje med hur algoritmen är tänkt att fungera. Metoden kan implementeras med små ändringar av befintlig kod. Det finns dock ett problem, tidsåtgången är O(n^m) där n är antalet celler och m antalet staketkoordinater. Den initiala kostnaden blir hög.

4 Diskussion

4.1 Hantering av koordinater

Hantering av koordinater var av mindre betydelse för det här projektet. Därför valdes enklast möjliga lösning, en statisk fil. Detta gör det enkelt att vid behov lägga till en ny inhägnad eftersom samtliga koordinater är samlade på en plats. Det hade varit möjligt att lägga koordinaterna i databas eller hämtat dem från en estern källa. Men detta hade medför merarbete utan att tillföra något till projektets målsättning.

Ingen transformering skedde till plana koordinator. Detta motiveras av att varken faktiska avstånd eller vinklar används utan koordinaterna kan betraktas som enhetslösa. Det är matematiskt komplicerat att korrekt transformera koordinater i grader till plana koordinater då jorden inte är en sfär. Jorden är något tillplattad vid polerna med en största diameter rund ekvatorn. Internt så omvandlas koordinaterna till heltal eftersom heltal är enklare att hantera en flyttal.

4.2 Val av algoritm

En rutnätsbaserad algoritm valdes då den är enkel att administrera. Den kan smidigt användas för att hantera flera områden då ett gemensamt koordinatsystem lätt kan upprättas. Varje cell i rutnätet är ett självständigt delområde. En cell har en uppsättning egenskaper och kan innehålla en eller flera sektioner av staketet. Det går enkelt och snabbt att koppla en given position till en cell.

Valet av algoritm skedde utan djupare analys av beteendet hos den mobila enheten vid positionering. Detta är en brist och ett alternativt och bättre arbetsflöde hade varit implementera all nödvändig kod för att positionera enheten innan valet av algoritm. Detta hade gett kunskap om hur viktig enhetens beteende är vid positionering. Jag hade inte tidigare utvecklat en applikation för positionering.

Vid implementation av algoritmen och med en växande kunskap hur positionering fungera i praktiken så blev det allt mer tveksamt om algoritmen verkligen var användbar. Därför avbröts arbetet med att implementera algoritmen för en djupare analys av arbetet i dess nuvarande tillstånd. Detta kan ses som en iteration där algoritmen analyserades på nytt med kunskap som framkommit under arbetets gång.

Med utgångspunkt från olika scenarion kring hårdvarans beteende så framkom det att den rutnätsbaserade algoritmen endast kan hantera ett område som är flera mil stort. Det beror dels på enhetens beteende och dels på att området delas in i celler där varje cell är ett delområde. Behovet av positionering är omvänt proportionellt till delområdets storlek. Kopplingen mellan delområde och cell är ett hinder för att reglera delområdets storlek.

För att algoritmen ska vara användbar så måste den kunna hantera ett betydligt mindre område. Ett stort område kan alltid delas upp i mindre områden vid behov men inte tvärtom. För att bättre reglera delområdets storlek är det bättre att helt frikoppla delområde från cell. Ett lämpligt mått för att bestämma ett delområde av maximal storlek är minsta avstånd till staket. Det finns inget enkelt sätt att bestämma detta avstånd med rutnätsbaserade algoritmen. Ett alternativ är att slå samman flera celler för att på så sätt skapa ett större område är möjligt men detta leder till ökad komplexitet utan någon uppenbar vinst.

Algoritmen är har en hög grad av komplexitet för ett litet litet antal koordinater. Detta kan accepteras om algoritmen typiskt användas till ett stort antal koordinater eller om algoritmen används till flera områden av varierade storlek.

4.3 Att positionera

Hur väl en enhet positionera beror av dels på vilka tekniker som finns till förfogande och dels beteendet hos själva enheten. Att hämta in en ny position är starkt kopplat till aktuellt operativsystem. I Android så är positionering samlat under ett gemensamt gränssnitt oavsett teknik. Men algoritmens syfte är att sätta en given position i relation till den virtuella inhägnaden. Från vilken källa den positionen hämtas saknar helt betydelse för funktionen av algoritmen. Positionen är anonym.

Vad som har betydelse för hur väl algoritmen fungera är enhetens beteende. Hur fort och korrekt den kan positionera. Enhetens beteende är en begränsande faktor. En enhet har en viss batteri kapacitet vilket i sin tur reducera hur ofta en enhet kan positionera. Det saknas information om enhetens beteende. Det är därför inte möjligt att göra en djupare analys av hur väl en typisk enhet positionera.

4.4 Nytt förslag till algoritm

Ett allvarligt problem med den rutnätsbaserade algoritmen är kopplingen mellan delområde och cell vilket begränsar minsta möjliga område för inhägnad. En indelning i celler reducera mängden data om inhägnaden till just den cellen. Ett bra mått för att upprätta ett delområde av maximal storlek är att utgå från minsta avstånd till staket. Detta kan inte göra på ett enkelt sätt med den rutnätsbaserade algoritmen så en alternativ metod behövs.

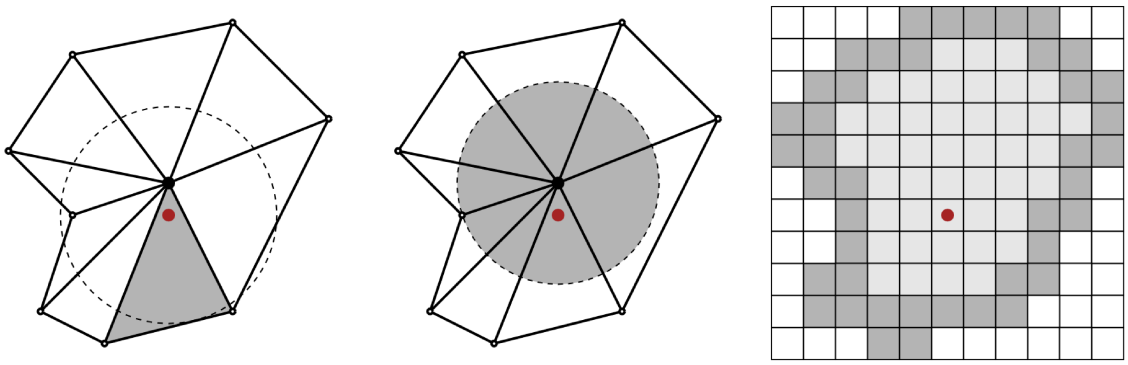

Bild 6: Vänster och mitten, genom att dela upp området i trianglar med hjälp av en fast fixpunkt så blir det lättare att bestämma avstånd och skapa delområde. Häger, förenklat rutnät utan stängseldata.

Det enklaste sättet att bestämma minsta avstånd till stängslet från en godtycklig punkt är att gå varvet runt från sida till sida. Detta ger O(n) där n är antalet sidor i staketet. Detta fungera för ett mindre antal koordinater men då den här operation måste utföras regelbundet och för varje område så är det önskvärt att reducera antal sidor.

Anta att området beskrivs av en konvex eller enklare konkav polygon istället för en godtycklig polynom. Då kan annan metod används för upprätta ett delområde där polygonen delas upp i trianglar som figur 6 visar. Från en fast fixpunkt så skapas en lista med avstånd och vinkel till samtliga staket koordinater. Listan med staketdata blir automatiskt sorterad under förutsättning att trianglarna inte överlappar varandra. Detta gör det matematiskt enkelt att bestämma vilken triangel som ockuperas av en given punkt, röd på bilden.

Figur 6 till vänster visas en given punkt som en röd punkt i en grå triangel. Avståndet mellan fixpunkten och den sida som utgör en del av polygonen antas vara minsta avstånd till staket. Ett delområde kan sen upprätta genom att rita en cirkel kring fixpunkten. Detta minskar totala antalet beräkningar.

Avståndet mellan fixpunkt och sida kan avvika från minsta avstånd till staket vilket framgår av den streckade cirkeln. Det är därför viktigt att den fasta fixpunkten runt vilken trianglarna upprättas väljs med omsorg. För extremt långsmala eller oregelbundna områden så fungera den här metod mindre bra. Fördelen är avståndet blir mindre ju närmare fixpunkten kommer staketet. Storleken på delområdet står i proportion till avståndet till staketet.

En annan metod är att skapa ett permanent inre område, bild 6 i mitten. Här en cirkel runt den fasta fixpunkten i centrum. Utanför det inre området kan triangelmetoden ovan användas. En annan metod för att skapa ett inre område är genom att reducera antalet koordinater från n till m och alltså från O(n) till O(m) för att bestämma minsta avstånd. För ett stort antal koordinater så ger detta en beräkningsmässig vinst.

Fördelen med ett rutnät att det förenklar arbetet med flera områden och gör det möjligt att dela upp inhägnaden i olika egenskaper. I bild 6 till höger så är inhägnaden uppdelad i innanför, staket och utanför. Den rutnätsbaserade algoritmen kan förenklas betydligt om ingen data om själva staketet sparas. Då behöver inte staketet delas upp i sektioner för varje cell. Den triangelbaserade metoden komplettera den rutnätsbaserade.

4.5 Administration av inhägnad

Detta arbete utgick från att inhägnaden skulle administreras på den mobila enheten. Det innebär att den mobila enheten från en uppsättning koordinater skaparen inhägnad och regelbundet kontrollera enhetens förhållande till den. Endast då enheten lämnar inhägnade informeras berörda parter. Men om enheten administreras centralt så kan det vara fördelaktigt att överlämna en del arbetet till en tjänst.

I det enklaste fallet så hämtar den mobila enheten koordinaterna till staketet och använder dessa till den virtuella inhägnaden. Metainformation kan skickas med koordinaterna. Det kan vara unika för just den enheten.

Inhägnaderna kan administreras centralt. När en enhet rapportera att den lämna sitt område så beräknas nytt delområde skickas till den aktuella enheten. Varje mobil enhet tilldelas ett område skräddarsytt efter deras aktuella position. Skillnaden mot tidigare är att mobila enheten endast känner till ett mindre delområde och inte hela inhägnaden. Fördelen är att mycket beräknings arbete sker utanför den mobila enheten.

Man kan också tänka sig att den mobila enheten endast rapportera in aktuell position. All hantering av inhägnad sker utanför den mobila enheten. Detta innebär dock en markant ökad användning av radio. Så någon vinst med denna metod finns inte.

4.6 Användningsområden för inhägnad

Motivet för att omgärda den mobila enheten med ett virtuellt stängsel var skydda den data som finns på mobilen samt skydda det eller de nätverk som den är kopplad till. Men det kan finnas andra skäl att omgärda en mobil enhet med en virtuell inhägnad. Andra användningsområden för en inhägnad är:

- **Person skydd:** Det finns många situationer där det är intresse att veta om en person avviker från ett bestämt område. Det kan vara en äldre person som valt att bo hemma. Anhöriga och andra berörda parter informeras om personen lämnat sitt hem. Den här typen av skydd finns redan men fördelen med en inhägnad är att personen inte behöver spåras så länge han eller hon är innanför inhägnaden. Det ger högre integritet.
- **Gruppera information:** Värden är uppdelad på områden och inte positioner. En virtuell inhägnad kan användas för relatera olika typer av information till en bestämd plats. Typiskt turistinformation. Att hämta information om viss plats kan ske helt automatiskt om användarens nuvarande position är känd.
- **Närvarokontroll:** Närvaro eller frånvaro av en person kan användas för styra funktioner. Detta förutsätter att personen alltid ha den mobila enheten med sig. Man kan tänka sig att närvaro krävs för en person ska kunna logga in på sin dator och passera vissa dörrar. Alltså ett verktyg för ökad åtkomstkontroll. Omvänt så kan frånvaro användas för stänga ned datorer och andra system.

– **Reglering av funktionalitet:** På många industrier och andra skyddade platser så är fotografering förbjudet. Detta är svårt att kontrollera då alla mobila enheter som regel innehåller minst en kamera. En virtuell inhägnad kan användas för att reglera kameran och andra funktioner på den mobila enheten. Det är möjligt att avaktivera kameran helt automatiskt.

5 Slutsats

5.1 Inhägnad av mobil enhet

Målsättningen för detta arbete var att finna en lämplig algoritm för upprätta och upprätthålla en virtuell inhägnad på en mobil enhet och testa den på ett faktisk system. En rutnätsbaserad algoritm valdes för maximal skalbarhet. Att bestämma om viss given position är innanför eller utanför inhägnaden går snabbt då all data är samlad i tabell. Ett rutnät lägger också grunden till att hantera flera inhägnader samt relationer mellan dessa.

Den virtuella inhägnaden är en uppsättning koordinater som omsluter ett geografiskt område. Algoritmen delar upp detta områden i ett rutnät där varje ruta är en cell. Varje cell är antingen innanför, utanför eller innehåller en del av staketet. En cell innehåller all den information som behövs för att bestämma om en given fixpunkt är innanför eller utanför staketet. En cell kan också om så behövs innehålla ytligare metainformation.

Följande problem framkom:

- För att algoritmen ska fungera som det är tänkt så behöver området vara tillräckligt stort. Varje cell behandlas var för sig och är ett självständigt delområde. Det är storleken på cellen som bestämmer behovet av positionering. Därför så måste cellen vara en viss storlek för att positionering ska ligga på en rimlig nivå. Man kan tänka sig att minska upplösningen, alltså minska antalet celler, på rutnätet för att reducera behovet av positionering men detta innebär att mängden data i varje cell blir större.
- Det är svårt att från en godtycklig punkt bestämma minsta avstånd till staketet. Med kännedom om avstånd till staket och enhetens hastighet så kan minsta tid för att lämna inhägnaden bestämmas. Avstånd till staket är därför ett viktigt mått för att reducera behovet av positionering. Men en cell innehålla bara data om just den cellen. Finns ingen del av inhägnaden i cellen så måste rutnätet sökas av för hitta närmast cell med en del av staketet. En uppdelning i celler gör det svårare att bestämma minsta avstånd till staket. Det enklare att utgå från hela staketet.
- Endast begränsad information finns tillgänglig om den mobila enhetens beteende vid positionering. Information om enhetens beteende behövs för att bättre bedöma en algoritms användbarhet och lämplighet vid positionering. Enhetens beteende beror av flera faktorer, hårdvara, mjukvara, miljö, metod för positionering och olika typer av felkällor. Det går att göra vissa antagande om vissa av dessa faktorer medan andra betydligt svårare att skatta. En mobil

enhet har till exempel en begränsad batterikapacitet och varje positionering kräver en viss mängd energi. Både faktorerna är okända och, i synnerhet, energi kostnad för positionering är svår att skatta.

Ett avgörande problem i det här arbetet är att algoritmen inte kan hantera ett mindre område. Den nedre gränsen för områdets storlek är viktigare än den övre eftersom ett större område kan delas upp i mindre. En uppdelning på celler gör det enkelt att koppla information till en given geografisk position men kopplingen mellan cell och delområde är ett hinder för att hantera mindre områden.

För att lösa upp kopplingen mellan cell och delområde så är det bättre att använda minsta avstånd till staket. Med kännedom om detta avstånd kan behovet av positionering minimeras. Men den rutnätsbaserade algoritmen gör det svårt att bestämma detta avstånd. Det är lättare att utgå direkt från hela staketet. Det behövs därför två algoritmer, en för att bestämma behov av positionering och en för att hantera geografisk information.

Svårigheten med att hitta information om den mobila enhetens beteende är också det ett hinder för att bedöma en algoritm. En viktig fråga är hur ofta en mobil enhet kan uppdatera sin position och hur mycket detta kostar i energi. Denna balans mellan förmåga att positionera och energi konsumtion bestämmer hur ofta en enhet i praktiken kan positionera. En annan fråga är hur väl en enhet positionera, hur nära enheten är den sanna positionen. Osäkerhet vid positionering påverkar gränsområden som när exakt enheten är utanför inhägnaden.

Den valda algoritmen bedöms fungera väl för att hantera geografisk information samt relationer mellan inhägnader. Saknas koppling mellan cell och delområde så kan upplösning på rutnätet väljas godtyckligt. Detta är fördel vid hantering av geografisk information och annan platsbunden information. För hantering av staketet så behövs en annan algoritm. Det ska med den algoritmen vara möjligt att bestämma minsta avstånd till staket eller på annat sätt minimera behovet av positionering.

5.2 Förslag till fortsatt utveckling

Vid implementation av rutnätsbaserade algoritmen och utvärdering av denna så framkom det att mer grundläggande kunskap behövdes för att bättre bedöma en algoritms lämplighet. Två områden av intresse är enhetens beteende vid positionering samt att bestämma aktuell position så noggrant som möjligt. Dessa tillsammans

är kopplat till enhetens beteende men vad kan göras för att kompensera för brister vid positionering. Dessa två pusselbitar är centrala vid upprättandet av en virtuell inhägnad och bör därför vara kända.

5.2.1 Den mobila enhetens beteende

Det kan antas att det skiljer mellan olika modeller och tillverkare. Att upprätta en bild över beteende hos mobila enheter vid positionering skulle vara till stöd vid skapandet av alla typer applikationer som använder positionering. På mobila enheter som använder Android så en

fältstudie enkel att genomföra. Demo applikationen för test av inhägnad kan skrivas om för att studera beteende vid positionering. Svårigheten ligger i att även mobila enheter som använder andra operativsystem behöver testas.

Följande parametrar är av intresse:

- **Batterikapacitet:** Hur mycket påverkas energi konsumtionen då GPS och andra metoder för positionering används. Finns det ett linjärt samband mot antalet positioneringar. Att ett linjärt samband råder kan inte antas då positionering i hög grad påverkas av hur den utförs. Det går att till exempel att samordna positionering med andra applikationer.
- **Tid för att etablera en position:** Hur lång tid tar det för enhet att etablera en position första gången? Detta är av mindre betydelse. Har positionering varit aktiverat en längre tid på den mobila enheten så finns i regel en position tillgänglig. Det finns alltså en position att fall tillbaka på vilket tillsammans med död räkning kan användas för skatta nuvarande position.
- **Tid för att uppdatera en position:** Hur ofta kan en mobil enhet uppdatera sin position? Detta är av stor vikt ty detta bestämmer upplösning. Denna faktor är viktig för att kunna följa en mobil enhet i rörelse. I detta arbete så sätter detta gränsen för minsta möjliga inhägnad.
- **Graden av precision vid positionering:** Hur exakt är en given position. Precision anges som en cirkel runt en given position inom vilken den mobila enheten faktiskt befinner sig. Storleken på cirkeln variera med olika typer av positionering. Detta kan vara av intresse att jämföra mot en GPS-navigator eller alternativt använda lantmäteriets fasta fixar. Ett exempel på en fast fix är en i marken nedsänkt kopp täckt av triangel markerat lock. Deras läge är uppmätt med mycket stor precision.
- **Hur mycket påverka den fysiska miljön:** Hur mycket påverkas den mobila enhetens förmåga att upprätta en positionering i olika miljöer? Satellitpositionering fungera till exempel mycket dåligt inomhus. Andra faktorer som kan påverka är närliggande byggnader, flervägsfel, och väder.

5.2.2 Att bestämma position

Det är ett icke trivialt problem att bestämma den mest troliga positionen utifrån tillgänglig information. Det finns flera orsaker, en positionsangivelse är alltid behäftad med en viss osäkerhet. Granden av osäkerhet beror bland annat på positionsleverantör och olika felkällor. Genom att använda flera positionsangivelser så kan graden av osäkerhet minskas.

Följande kan användas för att förbättra positionering:

- **Upprepad positionering:** För att skatta nuvarande position så används flera positioner. Detta förutsätter att den mobila enheten har varit i vila tillräckligt länge för ackumulera ett antal positioner. Det total felet kan minska under förutsättning att det inte är systematiskt.
- **Flera positionsleverantörer:** På en mobil enhet finns det tillgång till flera positions leverantörer. Man kan tänka sig att dessa används tillsammans för att minska felet och öka noggrannheten vid positionering. Det är dock tveksamt om detta ger förbättrad möjlighet att bestämma aktuell position. Dels finns ingen möjlighet att bestämma att positionering ska ske med olika tekniker vid en och samma tidpunkt och dels så finns endast begränsad information om positionsleverantörerna, vid programmering.

- **Hastighet och riktning:** Hastig och riktning kan användas för att med död räkning skatta nuvarande position. Förutom position så registreras alltid tiden för en positionering. Det är därför möjligt att skatta riktning och hastighet vid given tidpunkt.
- **Använda sensorer:** En mobil enhet har sensorer för att detektera rörelse, miljö och läge[13]. Men vilka sensorer som finns tillgängliga variera med hårdvara och operativsystem. Det är möjligt att bestämma om en mobil enhet är i vila eller inte. Rörelseriktning kan kompletteras med kompassriktning.
- **Dynamiska områden:** Genom att samla data under längre så är det möjligt att fastställa om användaren finns inom ett begränsat område. Detta kan användas för att skapa en ny inhägnad anpassad efter användarens rörelsemönster. Detta minskar det administrativa arbetet och reducera behovet av positionering.

13 *Android Developers (u.å.) Sensors Overview [Internet] Från: developer.android.com/guide/topics/sensors/sensors_overview.html [Hämtad: 2 okt 2013]*

6 Referenser