HÖGSKOLAN
I GÄVLE

AKADEMIN FÖR TEKNIK OCH MILJÖ
Avdelningen för industriell utveckling, IT och samhällsbyggnad

Algoritm för lokalisering av referensnoder

Indoor Positioning System

Selma Abbassi
Rickard Engström

2016

Examensarbete, Grundnivå, 15 hp
Datavetenskap
Dataingenjörsprogrammet
Examensarbete för högskoleingenjörsexamen inom datavetenskap

Handledare: Åke Wallin
Examinator: Jonas Boustedt

Algoritm för lokalisering av referensnoder med Indoor Positioning System

Selma, Abbassi
ndi13sai@student.hig.se

Rickard, Engström
nbt09rem@student.hig.se

21 juni 2016

Akademin för teknik och miljö
Examensarbete i datavetenskap 15hp
Högskolan i Gävle vt16

Handledare: Åke Wallin
Examinator: Jonas Boustedt

Abstrakt

Indoor Positioning Systems lokaliserar människor och objekt inomhus med hjälp av minst tre kända referenspunkter. System för inomhuspositionering som använder kända referenspunkter kallas anchor-based lokalisering medan de som beräknar deras positioner själva kallas anchor-free lokalisering. Syftet med detta arbete är att utveckla en algoritm som är anpassad efter ett anchor-free lokaliseringssystem. Den ska vara oberoende av nätverksuppkopplingen, hårdvaran och hur avstånden mellan mottagare och sensorer beräknats. Utgångspunkten för algoritmen är enbart avstånd mellan en mottagare och tre sensorer vilket kan beskrivas som arbetets huvudsakliga problem.

Algoritmen implementerades i Java med en simulering som återspeglar positioneringen i en perfekt miljö och sedan testas på en Android-applikation. Simuleringen tillåter användaren att rita ut flera mätpunkter som skapar en rutt. Dessa mätpunkter utnyttjas för att dynamiskt lokalisera referenspunkterna och mätpunkterna genom att hitta ett minsta avstånd mellan sensorerna. Dessa avstånd kan beskrivas som sidorna för en referenstriangel som möjliggör att ett koordinatsystem kan spännas upp.

Resultatet av den empiriska studien visade en felmarginal mellan $0,3 - 6m$ utan signalstörningar, vilket inte var tillräckligt noggrant. Efter att algoritmen implementerats lades fokus på en korrigering som kan itereras igenom för att uppskatta bättre mätvärden för referenstriangeln. Korrigeringen gav positiva resultat med lägre felmarginal. Arbetet kan vidareutvecklas genom att implementeras i ett verkligt IPS-system och algoritmen kan förbättras genom att skapa utökade funktioner som kan hantera fler än tre beacons.

Nyckelord: Indoor Positioning System, Anchor-Free localization, beacon, algorithm

Författarnas tack

Vi vill tacka vår handledare Åke Wallin för allt stöd och all vägledning han givit oss under arbetets gång och examinator Jonas Boustedt för hans insiktsfulla sätt att driva studenter till diskussion och kritiskt tänkande. Vi vill även tacka kontaktansvariga på Syntronic Håkan Sjörling och Magnus Sandberg för att de givit oss chansen att få arbeta i Syntronics lokaler där vi fått tillgång till resurser och tagit del av personalens expertis. Vi vill även tacka dem för deras visionära idéer inom arbetet som lyft upp vår ambitionsnivå och drift. Slutligen vill vi tacka Robert Stewing och Sören Hector för deras engagemang i vårt arbete och för all hjälp i matematik de givit oss.

Innehåll

Figurer

Tabeller

Begrepp

Beacon är en sensor som skickar ut signaler med generell avsikt att dra uppmärksamhet till en specifik position. I detta arbete talas det främst om beacons som sänder radiosignaler exempelvis routers eller Bluetooth beacons.

Anchor node är en referenspunkt, kan även benämnas ankarnod i texten.

API står för Application Programming Interface eller applikationsprogrammeringsgränssnitt och är en specifikation av hur olika applikationsprogram kan användas och kommunicera med en specifik programvara, som vanligen utgörs av ett dynamiskt länkat bibliotek.

RSSI (Received Signal Strength Indication) är en mätning för effekten av den mottagna signalstyrkan, Received Signal Strength (RSS). RSS är det faktiska värdet för signalstyrkan som mottages. Skillanden mellan de två begreppen är att RSSI är ett relativt värde utan enhet som alltid är positivt medan RSS kan mätas i dBm, dB, mW eller W.

UDP (Undetected Direct Path) är ett tillstånd som uppstår när den mottagna signalstyrkan från en oblockerad väg mellan sändare och mottagare faller under den detekterade tröskeln för mottagaren men den totala effekten för signalstyrkan fortfarande är över tröskeln.

Kapitel 1

Inledning

Här introduceras inomhuspositionering med översiktlig beskrivning av områdets mest fundamentala aspekter. Bakgrunden beskriver varför arbetet utförts, syftet framställer uppdraget och vilka resultat den ska leda till och frågeställningarna klargör vilka problem som ska lösas. Kapitlet innehåller även avgränsningar som valts för att kunna utföra arbetet inom den givna tidsramen.

Den utökade användningen av mobila enheter samt dess snabba tillgång till trådlösa nätverk har ökat användningsområdena för lokalisering. Location-Based Service (LBS) är den generella termen för lokaliseringstjänster och tekniken är en stor del av människors vardag.

Användningsområden för LBS kan innebära applikationer för nödsituationer som kan avslöja positionen för en person i nöd, navigeringstjänster som förser med riktningar inom geografiska kartor, informationstjänster för närmsta affärer eller för en okänd stad och slutligen spårning- och förvaltningsstjänster, exempelvis spårning av postpaket [18].

Global Positioning System (GPS) är ett sådant system som används av människor över hela världen för navigeringshjälp. Den bygger på avståndsmätning med triangulering från ett antal satelliter. Eftersom satelliten skickar ut sin tidskod till mottagaren, kan denna beräkna avståndet till satelliten. Det krävs tre satelliter för att kunna utföra en triangulering. Utefter det kan mottagarens position räknas ut. Tekniken har fått stor uppmärksamhet bland allmänheten och användningsområdena är breda men för att uppnå liknande lokaliseringsmöjligheter inomhus krävs andra förutsättningar.

Detta leder till Indoor Positioning System (IPS) som är ett positioneringssystem avsett att lokalisera människor och objekt inomhus. Till skillnad från GPS måste signalerna kunna passera väggar, golv och andra hinder. System för inomhuspositionering använder därför radiosignaler, magnetiska fält, akustiska signaler eller annan sensorisk information som samlas in från mobila enheter [3]. Sändarna som tar emot dessa signaler kan således vara WiFi-routers, Bluetooth beacons eller LED-ljus [8]. Det krävs minst tre av dessa sändare för att

kunna bestämma mottagarens position. System för inomhuspositionering kan kategoriseras som ankarbaserade (anchor-based) eller ankarlösa (anchor-free) IPS-system. Ankarbaserade är sådana som har kända positioner på de tre referenspunkter som krävs medan ankarlösa lokaliserar referenspunkterna själva.

IPS har varit populärt i akademiska sammanhang i över tio år vilket har lett till ett flertal teoretiska resultat och positioneringssystem. Marknaden för tekniken har dock inte visat någon storartad framgång eftersom precision och noggrannhet har varit bristfälliga faktorer [7].

1.1 Bakgrund

Ankarlösa IPS-system har många fördelar eftersom algoritmer för positionering i ett sådant kan hantera okända miljöer. De flesta IPS-system är ankarbaserade men det finns även dokumenterade ankarlösa system som utgår från nätverksinformation för att beräkna avstånd. Fördelen med ankarlösa system är att beräkning av positioner inte påverkas av de manuellt förkonfigurerade ankarnoder som kan inneha felaktiga värden. Detta leder också till anledningen varför ankarbaserade system inte kan utökas lika enkelt som ett ankarlöst system då ett stort antal ankarnoder kan behövas för att definiera den okända arean [14].

Ambitionen är att skapa en mobil Android-applikation som kan positionera människor i valfria miljöer utan något beroende av nätverksinformation eller förutbestämda referensnoder. En algoritm bör därför framtas med fokus på att lokalisera okända referenspunkter i en inomhusmiljö för att sedan implementeras i en simulering. Algoritmen ska använda två väsentliga parametrar för att beräkna positionerna på sensorerna: ett minsta avstånd mellan tre sensorer som skapar en referenstriangel och riktningar som återspeglas av hur användaren har orienterat sig i området, detta ska beskrivas av en rutt som kan ritas ut i simuleringen. Algoritmen bör vara obereonde av vilka typer av sensorer som är utplacerade men även av nätverksinformationen då dessa två faktorer hindrar algoritmen från att användas generellt. Intentionen är att skapa en så generell algoritm som möjligt som är både teoretiskt och praktiskt obereonde av datakällan men även av hur den kan tillämpas senare.

För att verifiera algoritmen ska en simulering implementeras som presenterar resultatet med ett grafiskt användargränssnitt. Denna ska försöka förutbestämma prestandan i en verklighetsbaserad miljö med och utan signalstörningar. En empirisk undersökning ska utföras genom att skapa en Android-applikation som testas i en byggnad med verkliga förhållanden för att bekräfta att algoritmen kan hantera ett verkligt scenario.

1.2 Syfte

Huvudsyftet med arbetet är att utveckla en algoritm som enbart förlitar sig på ankarlös IPS-teknik för att lokalisera referensnoder vilkas positioner är okända

från början. Detta leder till att förutbestämda fasta punkter inte behövs och omplacering eller utökning av referenspunkterna blir således möjligt. En begränsning blir dock att algoritmen enbart kan utgå ifrån avstånd från mottagare till sändare.

Algoritmen ska skapas med ambitionen om att människor ska kunna vara oberoende av vilka typer av sensorer som finns i byggnaden. Användare ska befinna sig i en okänd miljö där algoritmen kan hantera både Wi-Fi routers eller Bluetooth beacons. Den ska även kunna hantera båda samtidigt. Ett annat krav för algoritmen är att den ej ska utgå ifrån nätverksinformation då tidigare ankarlösa system begränsat sig till detta. En fördel med att vara oberoende av nätverket är att algoritmen kan ha flera tillämpningar.

Det har implementerats olika arkitekturer för att bygga upp ett positioneringssystem inomhus och flertal algoritmer har framtagits men det finns ingen standardiserad teknik för en optimerad implementation då signalstörningar är ett återkommande problem inom IPS.

1.3 Avgränsningar

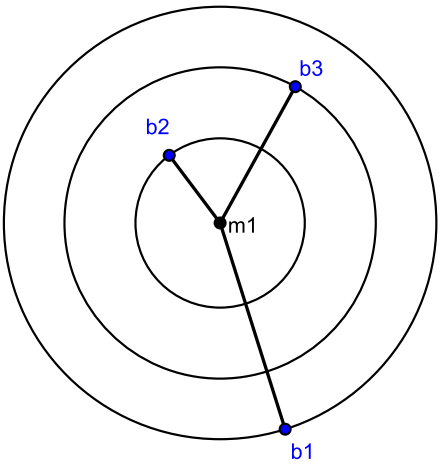
Arbetets huvudsakliga begränsning är att den kommer att anpassas efter ett ankarlöst positioneringssystem. Detta kan försämra noggrannheten samt ge bristfälliga teoretiska underlag.

Utvecklingen av algoritmen kommer att ske via en simulering som sedan testas med en Android-applikation. Beroende på hur bra resultat som visas i applikationen, med tanke på det allmänt kända problemet med signalstörningar, så kan arbetet ändra inriktning. Denna studie kommer inte att omfatta en förbättring av fysisk signalhantering i ett IPS-system.

Algoritmen kommer endast kunna hantera tre beacons som ger en referenstriangel och ett annat viktigt val är att begränsa programmet till en 2D-värld. Detta gör att algoritmens avståndsmätningar i verkligheten kommer ge felaktig information. Produktionskoden bör dock ha stöd för en 3D-baserad simulering som möjliggör vidareutveckling i framtiden.

1.4 Problemformulering och frågeställningar

Vanligtvis används tre referenssensorer med kända positioner för lokalisering inomhus. När avstånden mellan dessa är kända används trilateration för att finna användarens position. I detta arbete blir scenariot omvänt då utgångspunkterna endast är avståndet från mätpunkt till referenspunkter. Problemet kan beskrivas i Figur 1.1 där tre referenspunkter b_1, b_2 och b_3 kan befinna sig var som helst på sin respektive cirkel i förhållande till mätpunkten m_1 pga att enbart avstånden är kända.



Figur 1.1: Kännedom om enbart avstånd gör att sensorn kan befinna sig i en cirkel runt mottagaren

Avståndsmätning : Hur kan algoritmen finna en referenstriangel enbart utifrån avstånd? Hur kan avståndsdatat behandlas för att ge mer tillförlitliga resultat?

Algoritmens uppbyggnad kommer att ske inkrementellt och de två väsentliga inparametrarna är avstånd och riktning. Detta är den essentiella skillnaden från tidigare arbeten med tillägg att algoritmen inte ska behöva vara uppkopplad mot sensorernas nätverk.

Algoritmens uppbyggnad : Hur ser en algoritm för lokalisering av accesspunkter med IPS ut enbart genom att använda mottagarens avstånd till noderna samt riktningarna för orienteringsrutten som inparametrar?

Slutligen ska en empirisk studie utföras för att jämföra simuleringen med verkligheten vilket leder till ännu en frågeställning.

Algoritmen i verkligheten : Vilka faktorer är det som påverkar den empiriska studien jämfört med vad en virtuell simulering presenterar?

Kapitel 2

Relaterad forskning

Tidigare arbeten inom IPS beskrivs i detta kapitel. Studien omfattar olika tekniker som använts för att förverkliga ett inomhuspositioneringssystem samt härleder översiktligt de mest relevanta algoritmer som tidigare framställts.

System för inomhuspositionering har aktivt studerats i akademiska sammanhang samtidigt som de har begränsats på grund av praktiska hinder. Trots detta så har tekniken visat ett tydligt intresse bland allmänheten och det kanske bara är en tidsfråga innan den blir en del av människors vardag. Men vilka praktiska lösningar har forskningen kommit fram till och vilka är de specifika begränsningar som hindrar tekniken?

2.1 Användningsområden

När inomhuspositioneringstekniken hittar en optimal arkitektur för uppbyggnaden av ett sådant system kanske den har potential att överträffa nutida lokaliseringstjänster. Waqar et al. framhäver en intressant parallell mellan inomhuspositionering och människors behov av IPS, nämligen att moderna byggnaders utökade storlek gör att människor spenderar 70% av sin tid inomhus [20]. Lokalisering med IPS kan därför appliceras på flertal områden. Det kan användas för nödsituationer för att lokalisera människor i nöd i en byggnad eller markera evakueringsområden på en karta [1]. Det kan tänka sig även vara användbart i köpcentrum för att hitta affärer. Affärerna i sig skulle kunna använda tekniken för att lokalisera produkter, eventuellt visa erbjudanden på en karta. Ett annat tänkbart alternativ är att hitta personer med en viss expertis för att förenkla kontaktmöjligheter. Till exempel ett system där personer kan söka på personalen efter yrkesspecialitet och därefter med hjälp av IPS hitta var dessa befinner sig.

På flygplatser, sjukhus, skolor eller andra större anläggningar kan det vara svårt att hitta sin destination. IPS kan förbättra människors orienteringsupplevelser inomhus och förenkla deras vardagliga aktiviteter. Alternativen är

oändliga och kan täcka många samhällsområden vilket förklarar det ökade forskningsintresset samt påvisar att det finns ett praktiskt behov av tekniken.

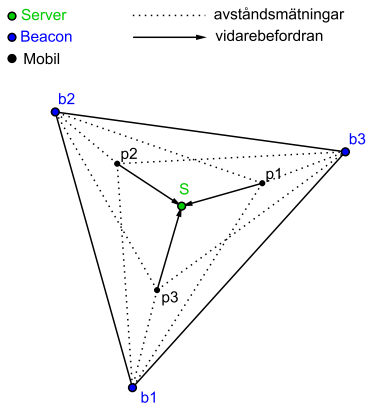
2.2 Ankarbaserad lokalisering

Det som är mest relevant när det gäller att jämföra denna studie med tidigare är att skilja mellan arbeten som utgår från kända positioner på sensorerna som ska användas som referenspunkter och de som beräknar dessa positioner själva. Ankarbaserad lokalisering kallas den förstnämnda kategorin medan den andra kallas ankarlös lokalisering (ankarbaserade lösningar är fortfarande intressanta för det här arbetet då olika kombinationer för avståndsmätning eller hårdvara används på samma sätt som i ankarlösa). Det som skiljer dem åt är hur avståndsmätningarna ska hanteras i algoritmen. Ett annat viktigt koncept inom avståndsmätning vid lokalisering är skillnaden mellan range-based och range-free lokalisering. Range-based handlar om lokaliseringsscheman som baserar sin uträkning på räckvidden vilket kan beräknas med exempelvis signalstyrkan medan range-free är scheman som behandlar nätverksinformationen för att räkna ut avstånd, exempelvis *hop count* mellan routers [9]. *Hop count* är antalet mellanliggande enheter (exempelvis routers) som datat måste passera från källa till destination. Enheten är således ett grundmått för avstånd i ett nätverk.

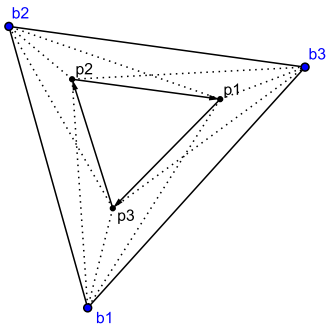
Algoritmen i detta arbete kan kategoriseras som en ankarlös och avståndsbaserad lokaliseringsalgoritm eftersom den baserar sina uträkningar på avståndsmätning och inte behöver ha några förplacerade fasta noder.

2.2.1 Centraliserade vs Distribuerade algoritmer

Det finns ett annat sätt att kategorisera algoritmer för inomhuspositionering. De kan antingen vara centraliserade eller distribuerade. Den första (Figur 2.1) in-



Figur 2.1: Centraliserad algoritm



Figur 2.2: Distribuerad algoritm

nebär att alla avståndsmätningar skickas till en central processor där uträkning för de uppskattade positionerna sker. Denna kan vara en server som visas i exemplet i Figur 2.1. En fördel med den centraliserade är att all information är åtkomlig i en enda nod. Den distribuerade lokaliseringen däremot (Figur 2.2) har ingen centrerad infrastruktur utan tillåter lokala noder att dela information för att själva räkna ut deras positioner [10]. Den distribuerade är att föredra i ett nätverk med ett stort antal noder då komplexiteten för uträkningen i centraliserade lokaliseringsalgoritmer ökar drastiskt om nätverket utökas [2].

2.2.2 Avståndsmätning

Eftersom GPS-signaler inte kan tränga igenom väggar krävs andra signaler för inomhuspositionering. Den allmänna lösningen är att använda radiosignaler såsom Wi-Fi eller mobilt nätverk då trådlösa nätverk är lättillgängliga och oftast inte kräver någon ombyggnad på deras infrastruktur. I en lokaliseringsalgoritm kan dessa radiosignaler utnyttjas på tre olika sätt för att beräkna avstånd mellan mottagare och sändare:

1. Angle of Arrival (AoA): För att bestämma avståndet med hjälp av AoA krävs det att enheten har flera antenner. Här är det viktigt att kunna uppskatta differensen i tiden mellan när signalen kommer fram till de olika antennerna. Med hjälp av avståndet mellan antennerna och ljusets hastighet är det möjlig att bestämma vilken vinkel signalen kommer ifrån.

2. Time of Arrival (ToA): ToA handlar om att bestämma avstånd genom att mäta svarstiden för en radiosignal från sändare till mottagare och sedan med hjälp av den hastighet som signalen färdats bestämma avståndet. Då radiovågor färdas i ljusets hastighet ($3 * 10^8 m/s$) krävs det att utrustning som kan bestämma tiden med samma höga noggrannhet är tillgänglig. Till exempel erhålls en noggrannhet på dm om tidsmätningen har en noggrannhet uppmätt i nanosekunder.

3. Received Signal Strength (RSS): Om signalstyrkan och frekvensen är kända parametrar är det möjligt att beräkna avståndet. En fördel med RSS är att det är lätt att få tag på informationen för dessa parametrar om Wi-Fi eller Bluetooth används. I WiFi behöver enheten inte ansluta sig till ett nätverk, utan kan med hjälp av det data som tillhandahålls när enheten skannar efter nätverk få den information som krävs för avståndsberäkning. En nackdel med RSS är att metoden är känslig för störningar.

Ytterligare ett sätt att beräkna avstånd på är att i Wi-Fi nätverk använda *hop count*, som beskrevs tidigare. Ett arbete som använder denna teknik kallar den för DV (distance vector) hop där ett range-free lokaliseringsschema utvecklats [16]. Den använder antalet hopp mellan routers som alternativ till det riktiga avståndet. En nackdel med DV-hop algoritmen är att noggrannheten för positionering är beroende av nod-densiteten.

Det mest förekommande problemet med inomhuspositionering är störningar som orsakar felaktiga mätningar vilket leder till dåliga empiriska resultat. Efter genomgång av befintlig litteratur i området kan konstateras att befintlig forskning fokuserar på att skapa simuleringar för verifiering av algoritmer som sedan jämförs med tester i en verklig miljö. Problematiken med signalstörningar uppmärksammas av alla studier men några av de beskriver även hur situationen kan förbättras.

Ett alternativ har föreslagits där RSS och ToA kombineras för att förbättra lokaliseringsprecisionen [11]. I studien menas det att ToA ej är tillräcklig som metod då sannolikheten för UDP (undetected direct path) ökar när avståndet mellan referensnoden och sensorn ökar. Noggrannheten försämras även drastiskt om signalen stöter på hinder. Algoritmen som framtagits använder en mobil ankarnod som förflyttar sig i rummet och använder RSS för att bestämma om accesspunkten är tillräckligt nära och om det finns hinder. Slutligen beräknas avståndet med ToA. Simuleringsresultatet visade att en RSS/TOA-hybrid sänkte medelvärdet på felmarginalen för positionering med 48.2% och standardavvikelsen med 71.3%.

2.2.3 Implementationer

För att förverkliga ett system för inomhuspositionering har olika kombinationer för avståndsmätning och positionering använts. Beroende på vilken typ av signal som ska hanteras (radiosignaler, magnetiska fält, ljud eller ljus) används olika typer av hårdvara.

Mobiler (främst smartphones) har inbyggda mätinstrument som kan utnyttjas för att underlätta lokalisering med IPS. Dessa inkluderar accelerometer, gyroskop och magnetometer, tillsammans kallade 9 Degrees of Freedom (9DoF). Accelerometern mäter acceleration och används oftast för att detektera antalet steg som tagits. Gyroskop mäter ändringar i orientering eller rotationshastighet och magnetometern mäter magnetiska fält som används som kompass.

Waqar et al. använder dessa verktyg med en kombination av Wi-Fi fingerprinting och ett Bayesian filter [20]. Wi-Fi fingerprinting är RSSI-baserad men skillnaden är att den sparar signalstyrkan från flertal accesspunkter samt klientens koordinater i en databas. Mätningar i realtid jämförs med tidigare sparat data ifrån databasen. Bayes filter är en algoritm som användes för att rekursivt uppskatta människans position utifrån databasens innehåll. Resultatet gav en felmarginal på avståndet som var mellan $4m$ och $6m$.

Ett liknande arbete som också tar hjälp av mobilens 9oF kombinerar Dead Reckoning (DR) och Wi-Fi fingerprinting [12]. DR använder tid, koordinater och riktningar från en kompass för att förutbestämma en position utifrån tidigare positionsmätningar. De experimentella resultaten visar att enbart DR gav bättre resultat än WiFi medan en kombination av båda sänkte felmarginalen. Bästa mätningen var 3 meters felmarginal. En kombination av Wi-Fi och Bluetooth har visat bra resultat där en felmarginal på $0.87m$ erhölls [4].

Ytterligare ett intressant område är robotteknik, som har visat tydligt intresse för IPS eftersom denna kan ha stor nytta av ett system som kan orientera

robotar inomhus [15]. Begreppet informationsstrukturerad omgivning tas bl.a upp, ett koncept som handlar om att göra omgivningen smartare istället för roboten. Ett välkänt exempel på en sådan omgivning är att ändra belysning i ett rum med kommandon. Denna aspekt är intressant för IPS då tekniken kanske behöver en helt ny anpassad informationsstrukturerad omgivning för att den ska utvecklas.

Alla ovan nämnda studier är några få av många kombinationer för att bygga upp ett IPS-system och de har alla en gemensam faktor: algoritmerna kräver att det finns kända positioner på minst tre referensnoder. Studier som presenterar algoritmer som inte är beroende av detta har benämnts som ankarlös lokalisering, vilket leder till nästa avsnitt.

2.3 Ankarlös lokalisering

Ankarlös lokalisering är lokaliseringssystem som inte kräver referensnoder med bestämda koordinater, vilket de flesta tidigare lösningar baserats på. Detta arbete är således en implementation av en ankarlös lokaliseringsteknik.

2.3.1 MDS-MAP

Tidigare arbeten med ankarlös lokalisering introducerades redan 2003 [17]. Här användes en centraliserad lokaliseringsalgoritm Multidimensional Scaling- MAP (MDS-MAP) för att finna minsta avståndet mellan noder enbart från nätverkets information. Metoden som användes kan delas upp i tre steg. Dessa beskrivs i artikeln:

> The method, MDSMAP, has three steps. Starting with the given network connectivity information, we first use an all-pairs shortest paths algorithm to roughly estimate the distance between each possible pair of nodes. Then we use multidimensional scaling (MDS), a technique from mathematical psychology, to derive node locations that fit those estimated distances. Finally, we normalize the resulting coordinates to take into account any nodes whose positions are known."[17, p. 201]

Eftersom denna algoritm bygger på en centraliserad beräkning försämras prestandan avsevärt när antalet noder ökar. Anledningen till detta är för att andra steget i algoritmen, applicering av Classical MDS (det finns även Metric, Nonmetric och Generalized MDS) utförs utan att använda kända positioner för referenspunkterna.

2.3.2 AFLA

I ett annat arbete från 2003 beskrivs Anchor-Free Localization Algorithm (AFLA) som består av två faser; den första fasen producerar en grafisk inbäddning

som liknar den originala inbäddningen och den andra fasen använder en optimering för att korrigera och balansera lokaliseringsfel [14]. I första steget väljs 5 referensnoder genom att beräkna *hop count* mellan routers. Detta innebär att systemet måste ha tillgång till nätverket, vilket denna studiens algoritm inte behöver.
Författarna tar även upp tre viktiga fördelar med ankarlös lokalisering:

> First, establishing anchors is a manual deployment task, and may be cumbersome. Second, the numerical stability of anchor-based approaches is questionable, since they give more weight to anchor position estimates, and errors in those estimates will have undue effect on the global solution. Finally, anchorbased approaches may not scale well, since to combat the instability described above, a large number of anchors may be required to configure an unbounded working area."[14, p.340]

2.3.3 CATL

En senare studie från 2013 behandlar samma problematik [19]. Här presenteras Connectivity-based and Anchor-free Three-dimensional Localization (CATL), ett system som anpassas efter storskaliga sensornätverk både i 2D- och 3D-miljö. Schemat använder sig endast av nätverksinformationen, så ingen förutbestämd referenspunkt är känd. Studien beskriver även nackdelen med att använda *hop count* för distansberäkning när mottagaren befinner sig i konkava regioner då det kortaste avståndet kan böja sig vilket resulterar i att det uppskattade värdet avviker avsevärt från det euklidiska avståndet. CATL-algoritmen består av tre huvudsteg och beskrivs i artikeln:

1. Fastställa kompletterande nätverksstrukturer.
2. Detektera notch nodes (noder i konkava regioner).
3. Multilaterationsbaserad lokalisering som undviker notch nodes.

En märkvärdig notering om CATL är att den ej är utvecklad efter IPS-system. Dess syfte är att lokalisera sensorer i ett sensor-nätverk och eftersom den använder hop-count så är den endast anpassad efter nätverk som använder sig av internetprotokollet.

Sammanfattningsvis kan konstateras att tidigare studier med ankarlösa noder skiljer sig från detta arbete då det strävar mot att vara helt oberoende av sensorernas nätverk för att öka algoritmens tillämpningsområden. Det är dessutom inte önskvärt att bli beroende av en specifik hårdvara, tvärtom ska IPS-systemet kunna använda vilka signaler som helst för avståndsmätning. En nackdel med MDS-MAP, AFLA och CATL är att de begränsar sig till att vara beroende av nätverksuppkopplingen.

Kapitel 3

Metod

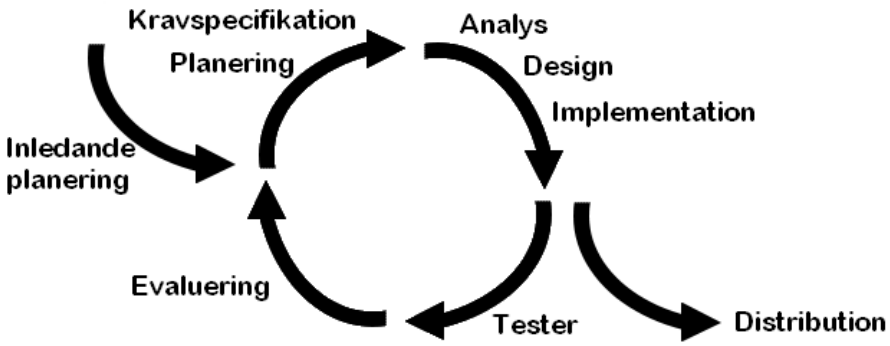
Kapitlet behandlar vilka arbetsmetoder för systemutveckling som applicerats för att sedan beskriva en utförlig analys av de mest väsentliga byggstenarna i en algoritm med IPS. Den egna algoritmen härleds utförligt där systemdesign och lösningar för påträffade problem beskrivs. Kapitlet innehåller även vilken datainsamling som valts för att öka noggrannheten samt beskriver hur den empiriska undersökningen utfördes.

Arbetet inleddes genom att studera området och dess tekniska aspekter för att få en bredare kunskap om inomhuspositionering. För att säkerställa att det inte publicerats tidigare undersöktes arbeten som behandlar ankarlös positionering. Med bättre förståelse för de teoretiska aspekterna gavs bra grund för att starta det praktiska arbetet. Arbetsmodeller och verktyg som var lämpliga att använda bestämdes och en grundlig undersökning av matematiska beräkningar som krävs för IPS utfördes.

3.1 Arbetsmodeller och processverktyg

Inom programvaruutveckling finns det fördelaktiga metoder såsom scrum och Test Driven Development (TDD). Dessa har använts då de förbättrar både källkodens prestanda och processens utveckling. Det metoderna har gemensamt är att de bygger på agila arbetssätt, att både individer och mjukvara ska vara anpassade för förändring. Detta leder också till det agila manifestet, en värdegrund som utvecklingen av arbetet inspirerats av (se appendix 8.1.1).

Algoritmens uppbyggnad utformades efter en evolutionär utvecklingsmodell. Detta innebär att lösningar för de problem som uppstått skapats inkrementellt. Till skillnad från vattenfallsmodellen så utförs arbetet på ett dynamiskt sätt. Denna arbetsmetod har ständigt kombinerats med en iterativ och inkrementell utvecklingsmetod som medför att funktioner i programmet har delats upp i mindre delar (inkrementellt) och utvecklats i upprepade cyklar (iterativt). Metoden illustreras i Figur 3.1.



Figur 3.1: Iterativ och inkrementell systemutveckling

Den evolutionära metoden kan påstås ha applicerats mer generellt på algoritmens uppbyggnad medan den iterativa metoden användes ständigt för mindre uppdelade funktioner i programvaruutvecklingen. Extreme Programming tillämpades, en metod som efterliknar den iterativa och inkrementella metoden då den delar upp processerna i delfunktioner. I Extreme Programming används bl.a parprogrammering, ett programmeringssätt som innebär att utvecklarna turas om att skriva koden på samma dator. En fördel är att en utvecklare koncentrerar sig på detaljer i koden medan den andra har en översiktlig kontroll över innehållet. Forskning har visat att detta ökar produktiviteten och förbättrar programvarans kvalitet.

3.1.1 Verktyg

Programvaran utvecklades med programmeringsspråket Java i utvecklingsmiljön IntelliJ 15. Ramverket JavaFX utnyttjades då den hanterar grafiska gränssnitt på ett smidigt sätt samt agerar modulärt mot andra plattformar. GitHub användes för versionshantering som är en viktig del i utvecklingsprocessen då det är viktigt att kunna spåra tidigare skapade funktioner samt återskapa dem.

3.2 Teori: algoritmens uppbyggnad

Utifrån tidigare framställda algoritmer har operationer som är de mest huvudsakliga för ett grundläggande IPS-system analyserats. Dessa inkluderar avståndsmätning och triangulering eller trilateration. Eftersom algoritmen måste anpassas efter ett system fritt från kända referenspunkter begränsas den till beräkningar som uppskattar värden och eftersom tidigare studier med ankarlös lokalisering kräver tillgång till nätverket bör algoritmen även kunna avstå ifrån det. Hur kan detta uppnås?

3.2.1 Mätpunkter

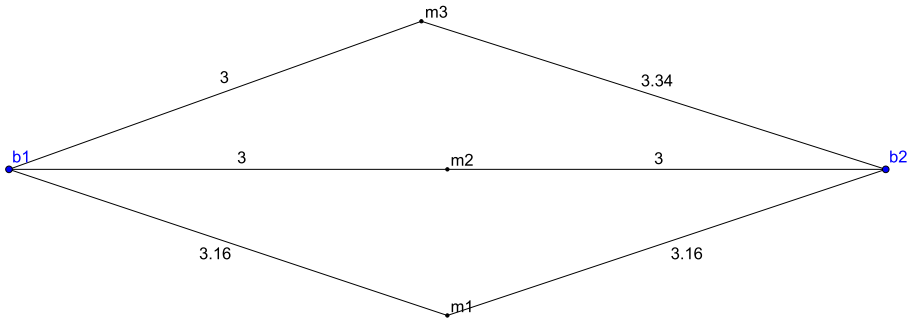
Mätpunkterna som tillsammans skapar en rutt bör innehålla ett avstånd till respektive beacon och en riktning. Det finns flera alternativa lösningar för att beräkna avståndet mellan sändare och mottagare som utgår från att använda information om egenskaper hos de signaler som skickas dem emellan. Denna information är mätdata som varje mätpunkt innehar. Algoritmen är oberoende av vilken metod som avses att användas, huvudsaken är att den kan ta emot dessa avstånd som inparametrar.

Avstånden ska vara angivna i meter och namn på sensorn (eller noden) är ett krav. Namnet kan vara enhetens BSSID eller så kan användare av systemet namnge sina utplacerade noder själva. Kortfattat kan förtydligas att varje mätpunkt har mätdata med information om enhetens namn och dess avstånd till varje sensor.

Den andra parametern, riktningen, är en vinkel mellan två vektorer. Den första vektorn är från första mätpunkten till en virtuell nordlig riktning i koordinatsystemet (y-axeln) och den andra vektorn är den från första mätpunkten till den nästkommande.

3.2.2 Avstånd mellan beacons

En lista med mätdatat skapades och traverserades för att finna de minsta avstånden till respektive beacon. Distansen mellan dessa beacons kunde då beräknas genom att summera två av de minsta avstånden som uppskattats. Det är med störst sannolikhet att det minsta avståndet mellan två beacons är det som uppmätts när mätpunkten befunnit sig på linjen mellan två noder.



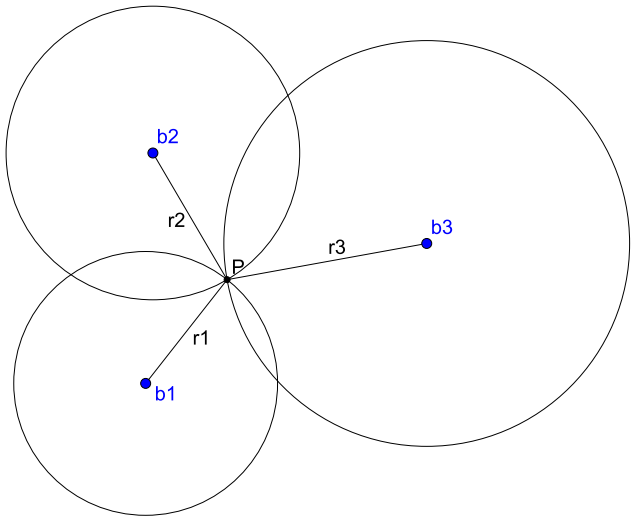
Figur 3.2: Minsta avståndet mellan två beacons

För att förtydliga ges ett exempel i Figur 3.2. Mätpunkterna m_1, m_2 och m_3 skapar tillsammans en rutt och beacons b_1 och b_2 är sensorerna. Även om det finns två liknande minsta avstånd $|b_1 - m_3|$ och $|b_1 - m_2|$ så är det summan $|b_1 - m_2| + |b_2 - m_2|$ som lagras som avståndet $|b_1 - b_2|$.

När detta även utförts på tredje referenspunkten b_3 med ytterligare mätpunkter erhålls tre minsta avstånd. Dessa skapar en referenstriangel vilkas vinklar beräknas med cosinussatsen. Med triangeln kan positionen för andra sensorer men även mobilenhetens position beräknas. Det senare alternativet kräver trilateration.

3.2.3 Trilateration

Trilateration innebär att mottagarens position beräknas utifrån skärningspunkten för de cirklar vars radie representeras av avstånden från de tre kända referensnoderna till mottagaren. Termen kan felaktigt förväxlas med triangulering men begreppen skiljer sig. Triangulering används när vinklarna är kända medan trilateration utgår ifrån avstånd.



Figur 3.3: Trilateration för att bestämma P

I Figur 3.3 visas tre beacons $b1$, $b2$ och $b3$. För att skapa ett koordinatsystem bör $b1$ sättas i origo och triangeln roteras så att $b2$ hamnar på x-axeln. Detta för att kunna definiera x-koordinaten för $b2$ som en variabel d vilket är det tidigare uträknade avståndet $|b1 - b2|$. Koordinaterna för b_3 är (i, j) och kan beräknas (via vinkeln γ och avståndet c som benämnda i Figur 3.5):

$$i = -\sin\gamma * c$$

$$j = \cos\gamma * c$$

Rotationen leder till att $P(x, y, z)$ kan beräknas enligt ekvationerna:

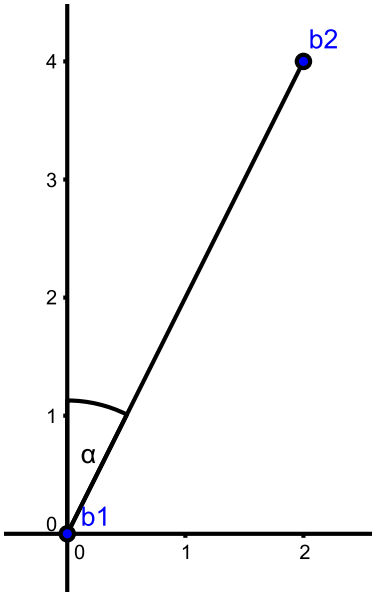
$$x = \frac{r_1^2 - r_2^2 + d^2}{2d}$$

$$y = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x$$

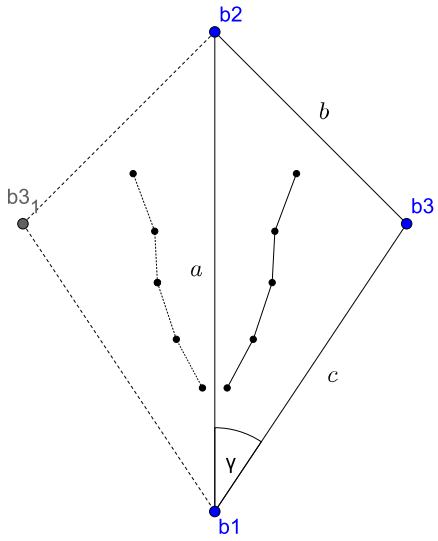
$$z = \pm\sqrt{r_1^2 - x^2 - y^2}$$

3.2.4 Rotation och spegling

Referenstriangeln har nu samma längder och vinklar som ursprungsnodernas triangel men dess orientering i det globala 2D-rummet är inte överenstämmande. Därför krävs det en referensriktning från $b1$ $[0, 0]$ till y-axeln $[0, 1]$ som gör att vinkelskillnanden mellan riktningen $[b_1, b_2]$ och referensriktningen kan beräknas. Referensriktningen är en virtuell riktning som skapas efter det uppspända koordinatsystemet som måste jämföras med den simulerade nordliga riktningen, och behövs för att finna hur många grader referenstriangeln ska roteras så att den överensstämmer med hur triangeln egentligen ser ut för noderna i simuleringen. Den simulerade riktningen motsvarar i verkligheten en uppmätt riktning från en kompass. Vinkeln för rotationen (α i Figur 3.4) beräknas genom skillnaden mellan referensriktningen och riktningen $[b_1, b_2]$. När α tagits fram kan en rotationsmatris appliceras.



Figur 3.4: Rotation



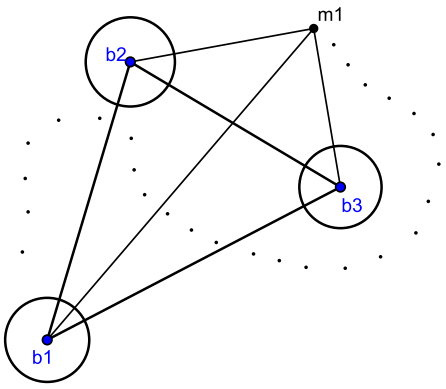
Figur 3.5: Spegling av b3

Ett problem som upptäcktes vid simulering var att den tredje punkten $b3$ speglades vid vissa tillfällen. Detta orsakas då algoritmen inte kan veta vilken av de motsatta riktningar (N eller S, Ö eller V) som tagits i förhållande till nodernas

positioner. Figur 3.5 visar ett scenario där en viss rutt tagits i triangeln och där b_3 . Avstånden från mätpunkt till sensorerna skiljer sig inte på någon av mätningarna för de två fallen vilket gör att algoritmen kan uppskatta triangelns position felaktigt. Algoritmen bör därför gå igenom listan med mätningar för varje rutt där den jämför första mätningens riktning med referensriktningen. Uträkningen utgår sedan ifrån första punkten för att avgöra riktningen för nästkommande mätning osv. Utförandet avser att jämföra den simulerade riktningen (eller kompassens) och den beräknade riktniningen för att fastställa om $b3$ behöver speglas eller inte.

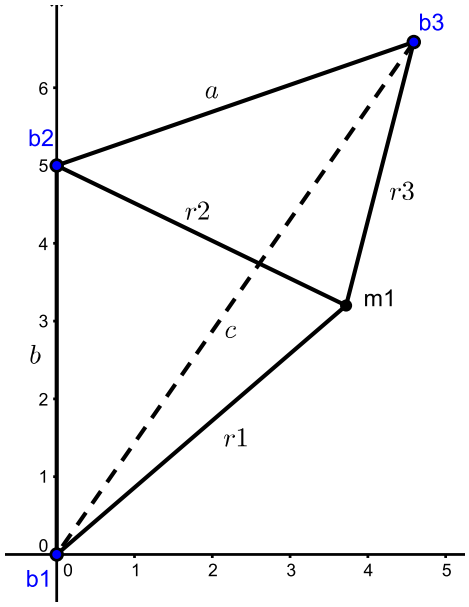
3.2.5 Korrigering

Ursprungligen var intentionen att mäta felmarginaler för algoritmen med Android-applikationen i en testmiljö men det visade sig att störningarna gav opålitliga resultat som inte hade förmåga att leda till någon bra slutsats. Den befintliga algoritmen optimerades och verifierades med en metod som approximerar referenstriangelns tre sidor genom att utnyttja alla mätpunkter.



Figur 3.6: Korrigering

Figur 3.6 visar en triangel med tre beacons och en rutt med mätpunkter. För varje mätpunkt i rutten erhålls tre avstånd till respektive sensor ($r1$, $r2$ och $r3$). Metoden utformades så att varje mätpunkt bidrar till en mer noggrann positionering av sensorerna genom att ett moln av punkter skapas för varje beräkning av positioner som ges av varje mätpunkt. Medelvärdet för molnet, som representeras av cirklarna på bilden, ger i teorin en bättre lokalisering än föregående metod då korrigeringen kan iterera igenom mätpunkterna n-antal gånger för bättre precision. Approximering av resultatet bör helst göras m.h.a minsta kvardratmetoden då den eliminerar avvikande värden. Metoden begränsades dock till en medelvärdesberäkning.

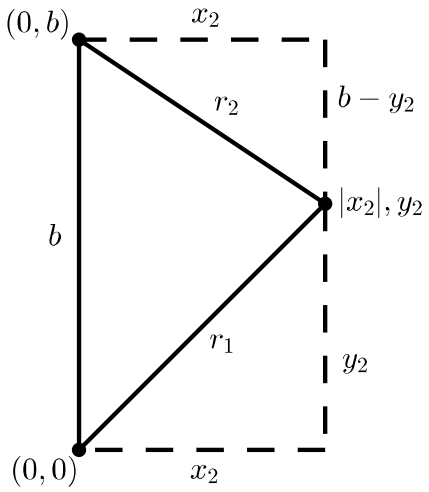


Figur 3.7: Positionering med Pythagoras sats

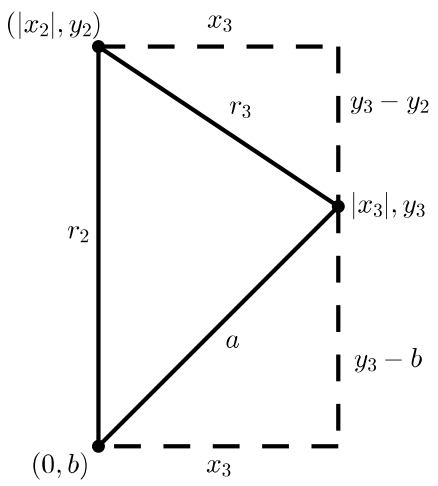
Ett förtydligande görs i Figur 3.7. Koordinatsystemet spänns upp genom att bestämma b_1 i origo och b_2 i punkten $(0, b)$. m_1 är mätpunkten och den streckade linjen är triangelns sida c som ska beräknas.

Positionering med Pythagoras sats

Två av avstånden mellan sensorerna som uppskattats innan korrigeringsmetoden används som utgångsvärden för denna metod. I Figur 3.8 är ett av de avstånden b och i Figur 3.9 avståndet a. Med Pythagoras sats ska avståndet c och tredje sensorns position uppskattas.



Figur 3.8: Lokalisera mätpunkten



Figur 3.9: Lokalisera tredje beacon

Figur 3.8 : Lokalisera mätpunkt (x_2, y_2) : Precis som i metod 1 sätts första sensorn i $(0, 0)$ och andra i $(0, b)$ för att spänna upp ett koordinatsystem. r_2 och r_1 är kända avstånd. Med Pythagoras sats kan $(|x_2|, y_2)$ beräknas med ekvationerna:

$$r_1^2 = x_2^2 + y_2^2 \Longrightarrow x_2^2 = r_1^2 - y_2^2$$
$$r_2^2 = x_2^2 + (b - y_2)^2 \Longrightarrow x_2^2 = r_2^2 - (b - y_2)^2$$

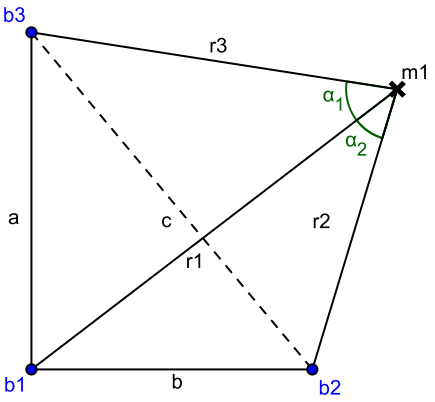
$$r_1^2 - y_2^2 = r_2^2 - (b - y_2)^2$$
$$b^2 - 2by_2 = r_2^2 - r_1^2$$

$$y_2 = \frac{b^2 - r_2^2 + r_1^2}{2b}$$
$$x_2 = \pm\sqrt{r_1^2 - y_2^2}$$

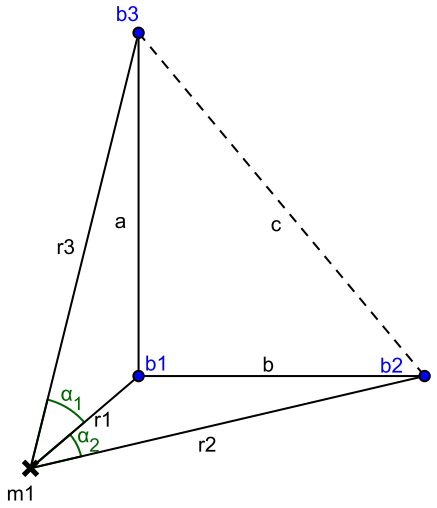
Figur 3.9 : Lokalisera tredje noden : a och r_3 är kända avstånd. Värdena sätts in i samma ekvationer som beskrevs i härledningen ovan men för att kunna göra det måste triangeln roteras så att den spänns upp i koordinatsystemet på samma sätt som visas i figuren.

Positionering med cosinussatsen

Ett problem med att rotera är att vinkeln antingen ska vara positiv eller negativ beroende på var mätpunkten befinner sig runt triangeln. Detta visade sig vara olösligt då mätpunktens position i förhållande till triangeln inte kan definieras. Detta dirigerade arbetet till en annan enklare metod där uträkning av c sker enbart med cosinussatsen. Vinkeln $\angle r_3 m_1 r_2$ kan beräknas för att med cosinussatsen avgöra c eftersom sidorna r_3 och r_2 är kända. Denna vinkel är summan eller differansen av a_1 ($\angle r_1 m_1 r_3$) och a_2 ($\angle r_1 m_1 r_2$). Figur 3.10 och 3.11 visar de fallen där $a_1 + a_2 = \angle r_3 m_1 r_2$.

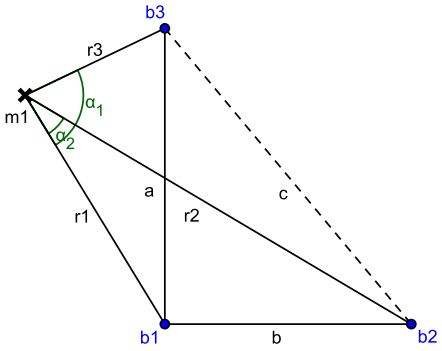


Figur 3.10: Fall 1

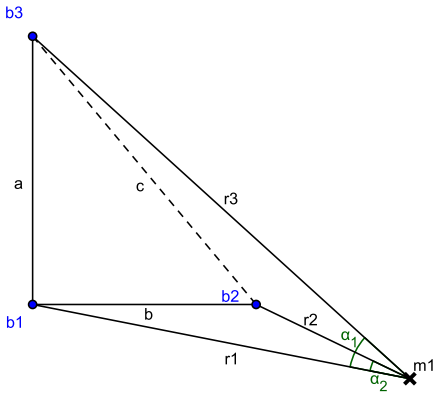


Figur 3.11: Fall 2

Figurer 3.12 och 3.13 visar de fall där $a_1 - a_2 = \angle r_3 m_1 r_2$.

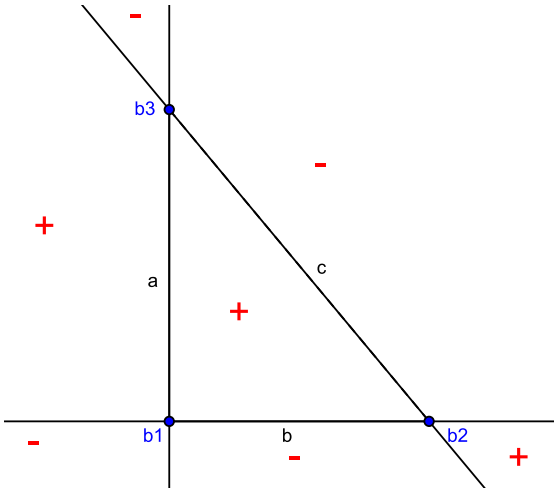


Figur 3.12: Fall 3



Figur 3.13: Fall 4

Ett problem som uppstår är att mätpunkten m_1 kan befinna sig i olika områden runt triangeln som gör att vinklarna a_1 och a_2 antingen ska summeras eller subtraheras och dessa gränsfall kan ej definieras. Gränsfallen illustreras i Figur 3.14.



Figur 3.14: Gränsfall

Denna problematik var samma anledning till varför positionering med Pythagoras sats inte fungerade när triangeln skulle roteras. Det går inte att definiera gränsfall utifrån var mätpunkten befinner sig runt triangeln på ett generellt sätt vilket i sin tur orsakar två olika fall. Positiv eller negativ vinkel i första metoden och addition eller subtraktion i andra metoden.

Slutligen hittades en lösning där båda fallen för varje mätning lagrades.

Tabell 3.1: Avstånd för c med båda fallen

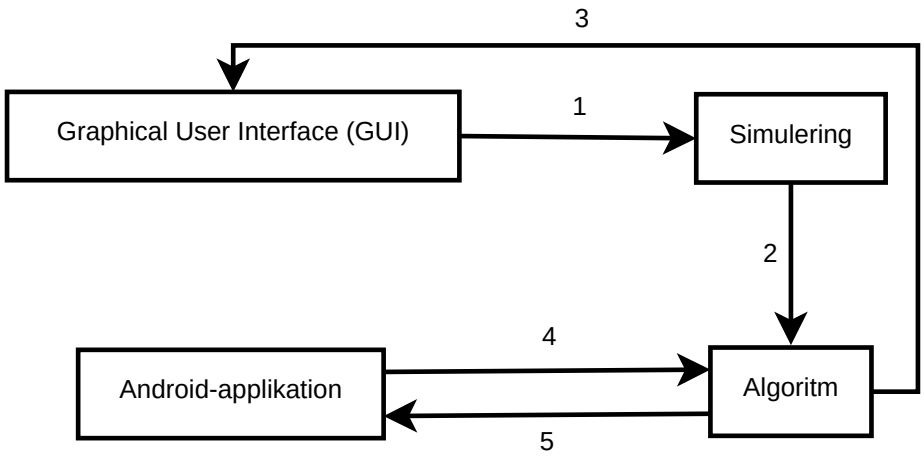
Avstånd på c	
$a_1 - a_2$	$a_1 + a_2$
10.1	11
15	10.5
10.3	17
21	10

Detta gav en tabell med värden för c som kunde analyseras och ge det mest förekommande värdet.

Varje rad bildar ett par av värden i tabell 3.1. Värdet som väljs som avstådet c är det som uppskattas som det mest förekommande värdet i paren. Metoden approximerar även värdena som är nära det uppskattade värdet. I exemplet i tabellen blir $c = 10,225$ som är medelvärdet av alla värden som är nära 10 (det mest förekommande värdet i paren).

3.3 Systemdesign

Två typer av implementationener konstruerades. Först utvecklades en Java-applikation i form av en simulering som visar att algoritmen fungerar i en störningsfri miljö men som även simulerar störningar för att ge ett mer verklighetsanpassat resultat. Den andra implementationen består av en mobilapplikation för insamling av verklighetsgrundat mätdata som faställer hur realistiska de simulerade störningarna är när de placeras i ett verkligt scenario. Systemets generella uppbyggnad evaluerades därför och delades upp i tre huvuddelar: Simulering, algoritm och Android-applikation. Systemschemat visas i Figur 3.15.



Figur 3.15: Systemschema

1. Användaren ritar ut en rutt i ett grafiskt gränssnitt eller lägger till störningar och utifrån dessa sker en en simulering av mätdata.
2. Det simulerade mätdatat skickas till algoritmen för beräkning av positioner.
3. Resultatet presenteras i användargränssnittet.
4. Den mobila enheten mäter upp avstånd till tre närmsta beacons. Datat skickas till algoritmen för beräkning.
5. Efter positionerna beräknats presenteras triangeln och användarens position på mobilskärmen.

3.3.1 Clean Code och hållbarhet

Bra programvara är sådan som är robust och hållbar. Med robust menas att källkoden inte påverkas av förändringar genom att exempelvis utöka programmet med nya funktioner eller klasser, utan har en stabil grund. Hållbar kod är sådan som kan återanvändas i framtiden och vidareutvecklas utan att befintliga ursprungsklasser påverkas. Ett sätt att tillmötesgå dessa två kvaliteter är att ständigt refaktorisera koden. Refaktorisering innebär att befintlig kod struktureras om på olika nivåer utan att dess externa beteende ändras. Ett exempel på refaktorisering är att skapa en egen klass `ClockwiseRotationMatrix` för rotationsmatrisen då matematiska formler lätt blir oläsbara vilket kan orsaka svårigheter vid utveckling. Genom att systematiskt kontrollera källkodens innehåll med målet att utföra refaktorisering, som förbättrar applikationens uppbyggnad, har programvara som är robust och hållbar skapats [13].

Ett annat sätt för att tillhandahålla detta och utveckla återanvändbar kod har varit att utforma kopplingar mellan klasser via objekt istället för primitiver. Fördelen med detta är att innehållet blir lättläst vilket ger utomstående läsare större förståelse för koden. För att skapa en mer modulär applikation separerades produktionskoden för algoritmen, det grafiska gränssnittet och simuleringen. Detta tillhandahölls genom att tillämpa MVC, ett designmönster som delar upp hela systemet i tre delar. Model innehåller domänens mest fundamentala klasser, View visar innehållet och möjliggör en interaktion och Controller förbinder Model och View genom att hantera kommandon som skickas av användaren via användargränssnittet.

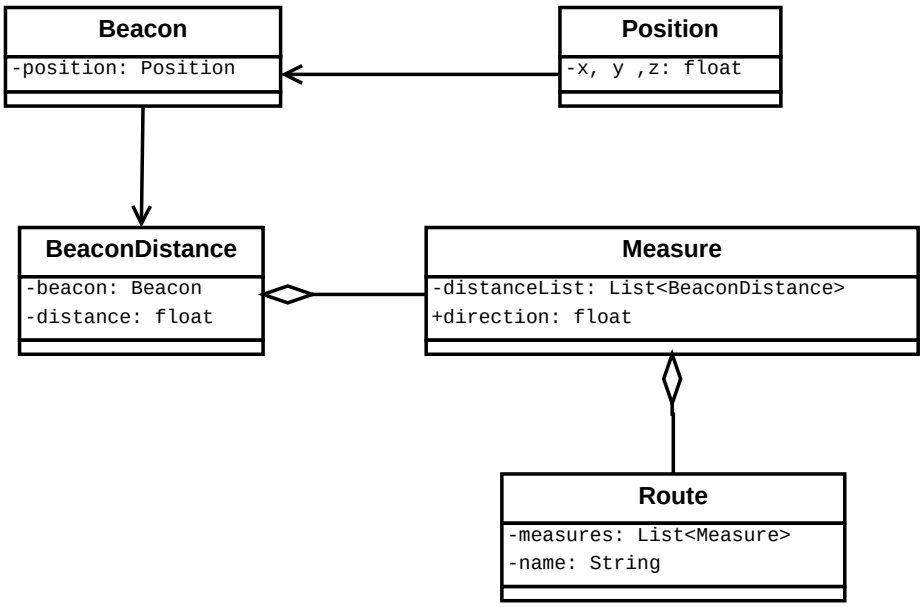
För att minska risken för oönskade buggar så har immutable-objekt prioriterats. Dessa objekt kan ej ändra tillstånd efter att de skapats och är även trådsäkra, vilket säkerställer programmets utformningar och beteenden. När immutable-objekt inte varit lämpliga att använda har den delen av koden inkapslats och enbart anropats genom andra klasser. Objektets tillstånd ska nämligen helst vara opåverkat av andra delar i programmet. Med detta som grund har immutable-objekt använts i alla de fall där det varit möjligt, utan att försämra strukturen eller påverka funktionaliteten. Immutable har även använts som substitut för att tillämpa designmönstret Prototype. Eftersom objekt av immutable karaktär inte behöver klonas utan kan använda sig av samma referens så skapar den ett nytt objekt när tillståndet förändras vilket efterliknar Prototypemönstret.

Projektstrukturen har uppfyllt the Single Responsibility Principle som är en av fem principer i S.O.L.I.D (se appendix 8.1.2). Principen förespråkar att varje klass inte ska ansvara för mer än en uppgift. På så sätt kan ett system byggas upp med högre hållbarhet. Klasserna har således delats upp i objekt som enbart är databärare, objekt som hanterar beteenden av olika slag och slutligen klasser som ansvarar för strukturella avseenden. För att skapa en sådan pragmatisk lösning representerades vanligt förekomande byggstenar med enstaka objekt. Exempel på sådana klasser är mätningar (`Measure`) och beacons (`Beacon`) som är två grundläggande klasser i programmets modell (Model).

3.3.2 Del 1: Algoritmen

Algoritmen skapades inkerementellt där matematiska problem löstes vartefter de uppstod. Första steget var att identifiera de parametrar som algoritmen kunde utgå ifrån. Avstånd var en grundläggande och självklar parameter för att finna en referenstriangel men implementationen för att finna avstånd mellan sensorerna skiljde sig från de flesta tidigare arbeten, då detta arbete begränsat sig till ett ankarlöst IPS-system. Riktning och rutter var något som behövdes för att kunna avgöra hur människor har orienterat sig men även för att kunna rotera triangeln rätt på skärmen i förhållande till hur noderna är placerade i verkligheten. Intentionen var att använda sig av en mobils kompass för att mäta riktningar eftersom riktningen tillsammans med avståndet kan avslöja var en beacon befinner sig. När parametrarna hade bestämts, påbörjades en analys med hjälp av UML-diagram för de mest väsentliga klasserna och den

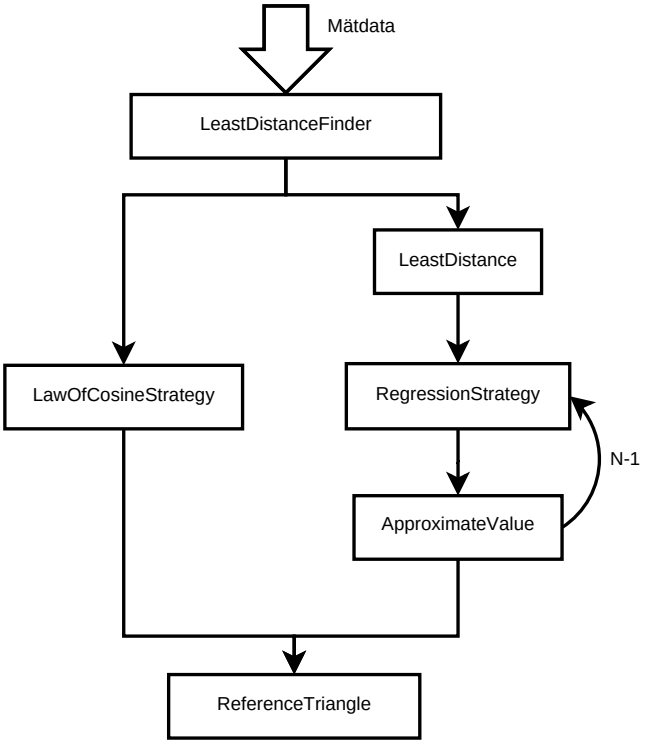
grundläggande strukturen för avståndsmätningen, som visas i Figur 3.16.



Figur 3.16: UML-diagram för avståndsmätningens grundstruktur

`Measure` är den klass som beskriver mätpunkterna. Mätningarna ska ha information om avstånden till alla noder samt en riktning från mobilens kompass. En lista på mätningar skapar en rutt (klassen `Route`). UML-diagrammets uppbyggnad är inte så annorlunda från hur det ser ut i resultatet, vilket tyder på en bra analys.

Algoritmen fick två olika metoder för att beräkna en referenstriangel. Dessa kan beskrivas i Figur 3.17. Designmönstret Strategy användes för olika matematiska implementationer. Strategy definierar en familj av beräkningar för en algoritm och tillåter användaren att ändra algoritmens beteenden i realtid. Eftersom mönstret kapslar in varje strategi så kan beteenden inom samma familj bytas ut mot varandra [5].



Figur 3.17: Arkitektur för algoritmen

Metod 1

`LeastDistanceFinder` är den klass som hittar de minsta uppmätta avstånden till sensorerna, såsom beskrevs tidigare i algoritmens teoretiska uppbyggnad. Utifrån dessa minsta avstånd beräknas avstånden mellan sensorerna som lagras i ett `DistanceBetweenBeacons`-objekt som tar emot en mätpunkt samt dess avstånd från två av sensorerna.

```
public DistanceBetweenBeacons(Measure measure,
    BeaconDistance beaconDistance1, BeaconDistance
    beaconDistance2)
```

När alla avstånd är kända, beräknas vinklarna mellan dem med strategin `LawOfCosineStrategy`, en klass som implementerar ett interface `TriangleFinderStrategy`. Strategin för cosinussatsens uppgift är att skapa ett objekt `ReferenceTriangle`.

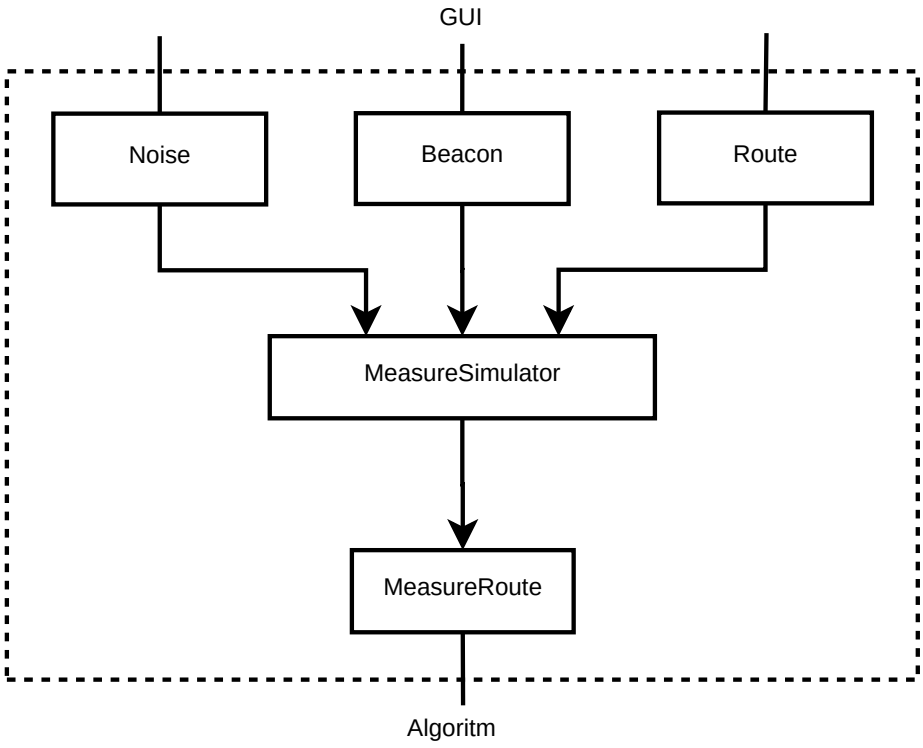
Korrigering

För att uppskatta värdena så bra som möjligt användes `LeastDistanceFinder` även i denna metod. Korrigeringen använder två av de beräknade sidorna a och b. Det är dock bara två sidor som behöver uppskattas då den tredje sidan kommer beräknas senare i processen. Implementationen för korrigering i algoritmen sker i klassen `RegressionStrategy` där ett objekt `RegressionList` skapas. Det är en klass som ansvarar för att bära det beräknade datat som krävs för att kunna utföra beräkningar såsom medelvärdesberäkning, minsta kvadratenmetoden och standardavvikelsen. Då varje beräkning får två lösningar från `RegressionStrategy` lagras de i `ValuePair`, en klass för att lagra parade data i form av Double. `ValuePair` ärver från `Pair` som är en generell klass för att kunna para ihop data av godtycklig typ. Den utökning som `ValuePair` har, förutom att den är specialiserad på att matcha två Doubles, är att det går att fråga dataparet vilket av dess två värden som är närmast. Denna funktion är nödvändig för att aproximationen ska fungera på ett smidigt sätt. Det lämpligaste sättet att implementera `RegressionList` blir då att via kompositon använda sig av en `ApproximateValue`. Dess uppgift är att lagra `ValuePair` och returnera de mest lika värdena i vardera par och på så sätt ge `RegressionList` de värden som behövs för att kunna utföra sitt syfte.

3.3.3 Del 2: Simulering

Simuleringen skapades via ramverket JavaFX som är en del utav Javas API. Valet av detta ramverk grundas i att det är ett modernt verktyg för att skapa grafiska gränsnitt. Större delen av applikationen har utvecklats med syntax som är kompatibel med Java 7 då de delar som har med själva algoritmen att göra även kommer att användas i en Android-applikation, medan de övriga delarna av källkoden nyttjar de verktyg som Java 8 har att erbjuda.

Simuleringens huvudsakliga uppgift är att på ett så realistiskt sätt som möjligt återspegla verkligheten. Den ska kunna justera störningsparametrar men även ha möjlighet att göra mätningar i en miljö fri från dessa simulerade störningar.

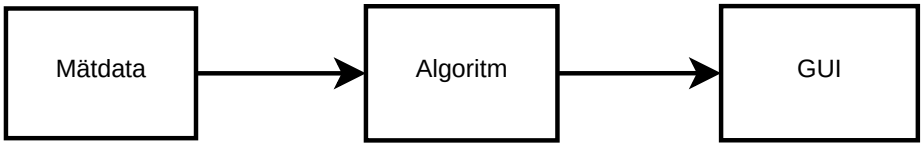


Figur 3.18: Arkitektur för simuleringen

Figur 3.18 illustrerar arkitekturen för simuleringen. Via användargränssnittet läggs `Noise` till av användaren som även ritar ut en `Route`, medan `Beacon` -objekt placeras ut slumpmässigt på skärmen. Det simulerade datat behandlas i en `MeasureSimulator` som skapar en `MeasureRoute`, en subklass till `Route<Measure>` som är en lista med mätningar. Detta är alltså simulerat mätdata som skickas vidare till algoritmen för beräkning och lokalisering av referenstriangeln.

3.3.4 Del 3: Android-applikation

I ett evaluerande syfte planerades utvecklingen av en Android-applikation som ska kunna bedömma hur realistisk simuleringen är i förhållande till verkligheten. Då signalstyrkan är enklast att använda på androidenheter kommer fokus att ligga på att använda signalstyrka även om algoritmen i sig inte har något utomstående beroende utav vilken typ av enhet som används eller hur dessa mätningar räknas ut.



Figur 3.19: Arkitektur för Android-applikationen

Figur 3.19 illustrerar en enkel beskrivning av Android-applikationens huvudsakliga uppgifter. För att beräkna avstånd utifrån signalstyrka används formeln för *free space path loss* där det krävs att både signalstyrka och frekvens är kända. Formel för *free space path loss*:

$$distance = 10^{27.55 - \frac{20 * log10(frequency) + signalLevel}{20}}$$

Genom att hämta information från telefonens avskanningar av WiFi-nätverk kan dessa två parametrar finnas. Informationen kan dock vara utdaterad och därför krävs det att en uppdatering av det insamlade datat triggas. I varje mätning använder telefonen batteriet. För att reducera batteriförbrukningen är det lämpligt att endast utföra nya mätningar när telefonen har förflyttats. Detta går att uppfylla genom att använda androids inbyggda sensor för accelerometern. Med accelerometern kan en stegräknare skapas som triggar en tråd när en mätning utförs.
På liknande sätt som i simuleringen ska mätdatat som hämtas från avståndsmätningarna presenteras visuellt på android-mobilens skärm. Estimeringen av mätdatans relativa position gentemot skärmen sker med hjälp av algoritmen då den anpassar sig till mobilens bildupplösning.

Den empiriska undersökningen visade tyvärr dåliga resultat vilket ledde till den alternativa metoden som nämndes tidigare (3.2.5 Korrigering). I arbetet optimerades därför algoritmen och slutligen jämfördes de två olika metoderna.

3.4 Datainsamling

Algoritmen kräver en stor mängd data för att uppskatta exakta positioner. En lämplig metod är att anpassa applikationen för crowdsourcing som innebär att allmänheten får ta del av ett projekt och bidra till den. Denna är lämplig att använda eftersom ju mer data som samlas in desto större chans att användare har gått tillräckligt nära en sensor eller linjen mellan två sensorer, vilket ger det rätta avståndet. Rutter som innehåller information med kortare avstånd ger således en mer trovärdig uppskattning av nodens position. Crowdsourcing skulle även kunna lösa problemet med signalstörningar då datamängden som samlas in kan analyseras och uppgiva det rätta värdet genom att beräkna medelvärdet eller minsta kvadratmetoden på det mest förekommande värdet.

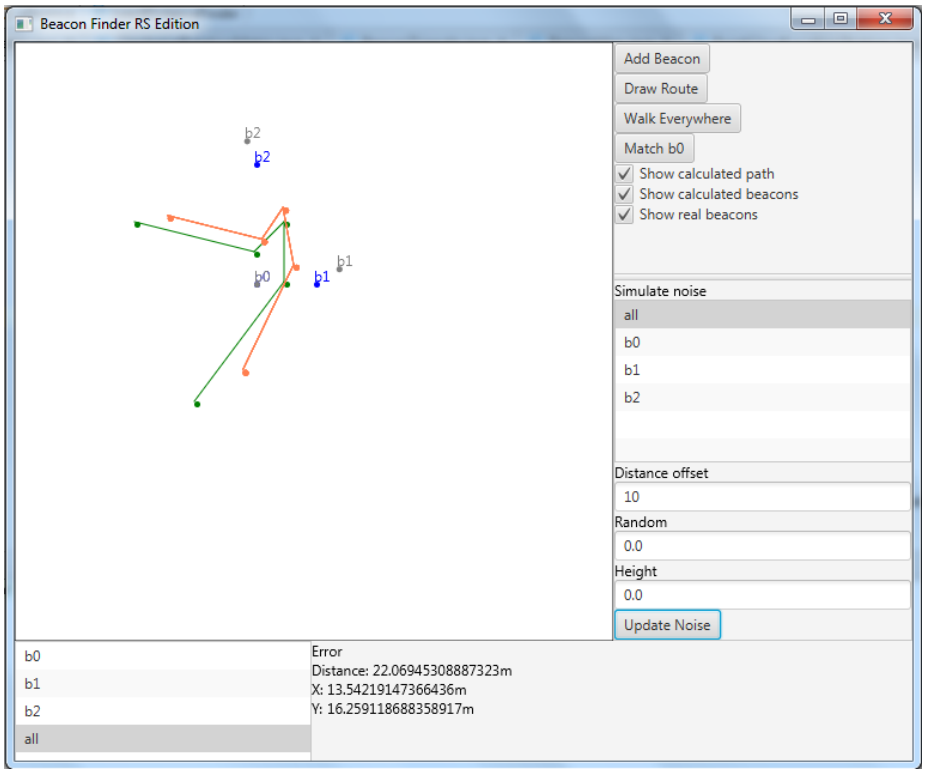
Den kvantitativa datainsamlingen bör skötas med hjälp av en server som kan hantera datamängden och algoritmens matematiska beräkningar. För att i framtiden kunna rita upp en karta på omgivningen skulle datat behöva analyseras för att uppskatta väggar och hinder.

Kapitel 4

Resultat

Kapitlet är en sammanfattning av algoritmen och simuleringen som slutprodukt. Tester som visar algoritmens prestanda och felmarginaler presenteras och möjliga förbättringar tas upp. Här besvaras även frågeställningarna från inledningskapitlet.

Algoritmen är oberoende av vilken signal som används för att mäta avstånd men även vilka typer av sändare som finns i närheten. Detta gör att den kan hantera både en centraliserad eller distribuerad nätverkstopologi beroende på hur algoritmen implementeras i verkligheten. Programvaran är öppen för vidareutveckling och kan därmed anpassa algoritmen för en utökning som hanterar fler än 3 beacons men även för en simulering i 3D. För att använda algoritmen i ett IPS-system bör simuleringen kunna rita upp en karta på omgivningen utifrån en analys av mätdata som lagrats i en server.



Figur 4.1: Skärmdump av simuleringen

Figur 4.1 visar simuleringens slutgiltiga version för detta arbete.

Blå punkt : en beacon som placeras ut slumpmässigt. Denna punkt är en sensors riktiga position som ska beräknas.

Grå punkt : en referenspunkt för en beacon som beräknats.

Grön linje : en rutt med mätpuntker som användaren ritar ut. Detta återspeglar hur användaren orienterat sig i verkligheten.

Orange linje : en beräknad rutt där användarens positioner (mätpunkter) tagits fram med trilateration.

4.1 Analys av resultat

Mohammed Reza Gholamis jämför algoritmer för inomhuspositionering i Wireless Sensor Networks (WSN) [6]. Författaren kom fram till att dessa algoritmer kan utvärderas på tre sätt. Den första mäter kostnaden för implementationen av algoritmen. Den andra mäter i procent antalet noder som kan lokaliseras,

bortsett från noggrannhet, och den tredje metoden är en kombination av flera kriterier för testfall.

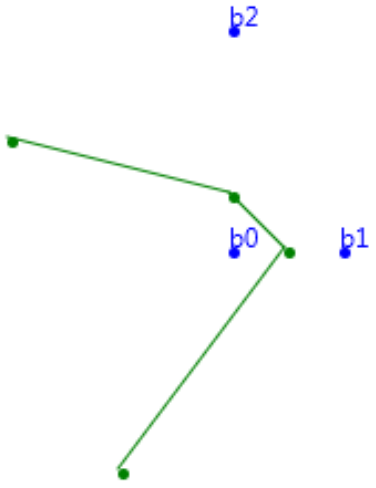
I den första metoden nämns bl.a algoritmkomplexitet (ordo-notation), antalet överförda paket i nätverket för att beräkna nodernas positioner och konvergenstiden som exempelvis kan beräknas genom att mäta hur lång tid det tar för lokaliseringen när nätverket utökas. Dessa metoder leder till hur denna algoritm kan evalueras.

4.1.1 Testfall

Två testfall skapades i simuleringen med målet att kunna jämföra algoritmen med korrigeringen. En beräkning av tidskomplexitet valdes bort då noggrannheten prioriterades framför algoritmens prestanda. Detta berättigas då fokuset lades på en teoretisk lösning som presenterades i en simulering där noggrannheten var väsentlig. Däremot hade prestandan varit minst lika betyldelsefull om algoritmen hade använts i ett verkligt scenario.

Algoritmen och korrigeringen fick nya benämningar där algoritmen kallades Metod 1 (M1) och korrigeringen Metod 2 (M2). Första testfallet beskriver en rutt som inte passerar en av triangelns sidor medan andra testfallet är en optimerad rutt som passerar mellan alla beacons. Motivationen till varför testfallen utformades på det sättet är att första metoden kan endast få det rätta minsta avståndet för triangelns sida om rutten passerat mellan två beacons. Testfallen kommer således visa tydligt skillnaden mellan M1 och M2.

Testfall 1

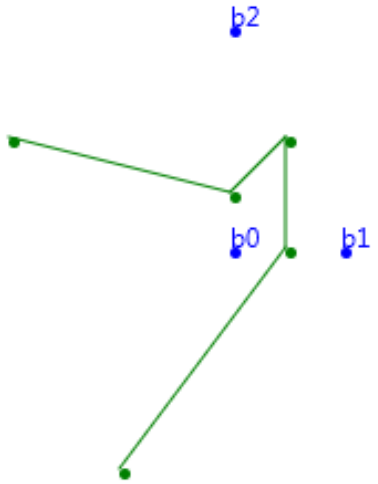


Figur 4.2: Simulering av testfall 1

Tabell 4.1: Mätningar för testfall 1

Mätningar	Störningar (m)			Metod	
	b1	b2	b3	M1 (m)	M2 (m)
m1	0	0	0	19.88	10.08
m2	2	0	0	15.94	9.36
m3	0	2	0	10.87	10.7
m4	0	0	2	12.89	11.33
m5	2	2	2	17.72	10.68
m6	10	10	10	24.13	28.82
m7	100	100	100	NaN	211.64

Testfall 2: optimal mätning



Figur 4.3: Simulering av testfall 2

Tabell 4.2: Mätningar för testfall 2

Mätningar	Störningar			Metod	
	b1	b2	b3	M1 (m)	M2 (m)
m1	0	0	0	0	0
m2	2	0	0	2.68	3.98
m3	0	2	0	1.71	1.21
m4	0	0	2	2.36	1.54
m5	2	2	2	4.13	4.44
m6	10	10	10	NaN	20.55
m7	100	100	100	NaN	248.17

4.1.2 Tolkning av testfall

Då testfall 1 (Figur 4.2) är valt för att visa de starka sidorna för M2 så tyder resultatet som väntat på att M2 ger bättre noggrannhet än M1. Även när störningar tilläggs fortsätter M2 ge lägre felmarginaler. En nackdel med M1 som framkommer i mätningen m_7 är att med tillräckligt stora mätfel går det inte längre att bilda en triangel vilket ger resultatet Not a Number (NaN) i Tabell 4.2 . Samma resultat uppdagas i testfall 2 på mätningarna m_6 och m_7 i Figur 4.3. Anledningen till att det inte går att skapa en triangel är att en sida uppskattas vara längre än de två andra sidorna sammanslagna eller att skillnaden mellan två sidor är större än den tredje sidan. Sammanfattat visar detta testfall att M2 är mycket mer lämpad än M1 när det kommer till ooptimerade rutter. Mätningen m_7 i testfall 1 är överraskande bättre än m_7 i testfall 2, detta kan förklaras med att approximationen av c inte är helt optimerad. Samma anledning kan även förklara varför m_1 i testfall 1 inte ger värdet 0, vilket den borde göra.

M1 och M2 ger bäst resultat i olika fall beroende på hur störningarna är balanserade. Det tyder på att M2 är mer känslig för störningar från vissa beacons då den nyttjar två av de tre referenspunkternas mätningar. Detta problem kommer att lösas när metoden itereras.

Styrkor och svagheter

Metod 1

- Styrka
  - Snabb och lätt att tillägga nytt mätdata.
- Svaghet
  - Känslig för var mätningarna utförs.

Metod 2

- Styrka
  - Ger mindre fel.
  - Mindre känslig för var mätningarna utförs.
  - Kan itereras för att få bättre resultat.
- Svaghet
  - Fler beräkningar än M1 vilket gör den långsammare.
  - Tillägg av nytt mätdata har inte hanterats.

4.2 Svar på frågeställningar

Hur kan algoritmen finna en referenstriangel enbart utifrån avståndsmätning?
Algoritmen är obereoende av hur avståndet mäts upp då arbetets fokus begränsats till teoretiska lösningar med grafisk representation via en simulering. För att finna referenstriangeln har minsta avstånden från en mätpunkt till två beacons summerats. Programmet hanterar `Route`-objekt som innehåller mätdata och riktningar. Om dessa rutter traverserar linjen mellan någon av sensorerna, dvs en av sidorna för referenstriangeln, så har det enda rätta avståndet mellan dessa noder hittats. En nackdel blir då att precisionen är beroende av ruttens utformning. När avstånden uppskattats används cosinussatsen för att beräkna vinklarna. Koordinatsystemet spänns upp utifrån triangeln vilket gör att alla mätpunkter och sensorer kan positioneras (med trilateration).

Hur kan avståndsdatat behandlas för att ge mer tillförlitliga resultat?
Till en början var avsikten att försöka finna bättre lösningar för de praktiska hinder som arbetet ställs inför i verkligheten med IPS. Den empiriska undersökningen var något som snabbt insågs skulle ändra syftet med arbetet avsevärt. En lösning för att förbättra de praktiska omständigheterna, dvs mer konkret studera signalbehandling, var inget område som låg i linje med arbetets syfte. Dessutom fanns det ingen ny kunskap från detta arbete som kunde erhållas i det avseendet. Detta var den största förändring som gjordes under

arbetets gång. Istället valdes en teoretisk verifiering av algoritmen, vilket skulle presentera ett resultat som var mer relevant för syftet. Testfallen visar att verifieringen förbättrade avståndsmätningen.

Hur ser en algoritm för lokalisering av accesspunkter med IPS ut enbart genom att använda mottagarens avstånd till noderna samt riktningarna för orienteringsrutten som inparametrar?
Algoritmens uppbyggnad för en mätpunkt kan sammanfattas i 8 steg:

1. Mäta avstånd från en mottagare till tre sensorer.
2. Anta avstånd mellan beacons för att uppskatta en referenstriangel.
3. Beräkna alla vinklar i triangeln med cosinussatsen.
4. Spänna upp triangeln i ett koordinatsystem genom att sätta $b1$ i origo och $b2$ i $(0, d)$.
5. Använda trilateration för att beräkna mätpunktens position i koordinatsystemet.
6. Rotera triangeln rätt m.h.a skillnaden mellan den virtuella nordriktningen och kompassens nordriktning.
7. Tillägga de fallen där triangeln kan spegla tredje noden.
8. Korrigering av avstånd mellan beacons.

En teoretisk korrigering utfördes för att verifiera avstånden i triangeln och implementerades sedan i simuleringen. Denna metod visade sig fördelaktig då den utnyttjar varje mätpunkt till skillnad från ursprungliga metoden som endast gav bra resultat vid rätt utplacerade rutter.
Korrigeringen kan sammanfattas i dessa steg:

1. Anta att $b1$ ligger i $(0, 0)$ och $b2$ ligger i $(0, b)$.
2. Anta att sidorna a och b på referenstriangeln från ursprungliga algoritmen stämmer.
3. Beräkna vinkeln $\angle r_3 m_1 r_2$ med cosinussatsen.
   (a) beräkna vinklarna a_1 ($\angle r_1 m_1 r_3$) och a_2 ($\angle r_1 m_1 r_2$).
   (b) addera och subtrahera a_1 och a_2.
4. Lagra båda fallen och beräkna avståndet c.
5. Analysera värdena och beräkna det mest förekommande värdet c genom att beräkna medelvärdet.
   (a) Medelvärdet bör fördelaktigt ersättas med minsta kvadratmetoden.

6. Återgå till steg 2 men använd sidan a och det beräknade värdet c för att bestämma b.

Vilka faktorer är det som påverkar den empiriska studien jämfört med vad en virtuell simulering presenterar?
Det som påverkade den empiriska studien var signalstörningar som orsakades av väggar, hinder, andra signaler i omgivningen och även det faktum att trådlösa routers kan ändra sin signalstyrka beroende på dess belastning. Frågan omformulerades därför för att kunna besvara huruvida korrigeringen var en lämplig metod att använda eller inte, eftersom arbetet ändrade inriktning.

På vilket sätt kan korrigeringen ses som en trovärdig verifiering? Korrigeringen använder två av triangelns sidor a och b som beräknats tidigare. Detta innebär egentligen ett antagande om att dessa avstånd är korrekta eftersom de kan innehålla felmarginaler, framförallt om algoritmen applicerats i verkligheten. Det som gör att korrigeringen är trovärdig är att den nyttjar varje mätpunkt genom att uppskatta avstånd via regression samt att den kan itereras igenom för att kontinuerligt få mer exakta värden. I dagsläget sker approximationen av avståndet genom att beräkna medelvärdet på alla uppskattade värden för c men en bättre metod är att använda minsta kvadratmetoden då den minimerar felen som uppstår.

Kapitel 5

Diskussion

Här tas viktiga förhållanden mellan arbetet och annan forskning upp samt hur resultatet bidrar till samhällets utvecklingsmål. Begränsningar och möjligheter till vidareutveckling diskuteras och jämförs med befintlig forskning.

Algoritmens allmängiltighet dvs förmåga att vara oberoende av hur avståndet mellan mätpunkt och referenspunkt beräknas, gör den öppen för vidareutveckling samt enkel att tillämpa. Simuleringen kan användas för övriga IPS-system där arkitekturen för systemet kan illustreras grafiskt och felmarginaler kan beräknas. En nackdel var oförmågan att tillämpa RSS i praktiken som gav bra avståndsmätningar vilket gjorde att fokus på arbetet hamnade på algoritmens teoretiska uppbyggnad.

Trots att algoritmen är skapad efter ett ankarlöst positioneringssystem har simuleringen potential att fungera som en verifiering för befintliga ankarbaserade IPS-system. Här menas det att med hjälp av programmet kan felmarginaler kontrolleras om fasta positioner på sensorerna bestämts från början och på så sätt kan standardavvikelser för sin egna positioneringsmiljö tas fram.

Syftet med arbetet var att skapa en algoritm för ett ankarlöst IPS-system som är oberoende av nätverksuppkopplingen till sensorerna men även av vilka sensorer som utnyttjas. Metoden som användes, dvs summering av minsta avstånden kombinerat med en korrigering som verifierar triangelns sidor, gav korrekta värden i simuleringen vilket påvisas i testfallen som utfördes i resultatet. Arbetets inriktning ändrades till en mer teoretisk uppbyggnad av en algoritm samt en simulering vilket betyder att slutprodukten inte kan benämnas som ett fulländat IPS-system men eftersom algoritmen är generell finns det goda möjligheter att implementera den i verkligheten.

5.1 Diskussion av litteraturstudie

Generella begränsningar som orsakas av alla ankarlösa system är att de har hög beräkningskomplexitet, låg noggrannhet för lokalisering samt bristfälliga teore-

tiska underlag. Den första avgränsningen har undvikits då matematiken bakom denna algoritm inte kan påstås inneha en hög komplexitetsnivå eftersom linjär algebra kombinerat med trigonometri tillämpats. En fördel med att använda den grundläggande matematiken är att beräkningarna inte försämrar algoritmens prestanda och är lättare att klarlägga för utomstående. Andra problemet, låg noggrannhet, är något som bekräftades vid den empiriska studien. Slutligen har bristen på tidigare implementationer av ankarlösa system givit upphov till att självständiga matematiska lösningar på problem som är specifika för detta arbete inte kunnat stödjas av forskning. Trots att algoritmen i teorin är oberoende av hur avståndsmätningen sker i verkligheten är denna en viktig aspekt att ta med i undersökningen. Hade Android-applikationen fått en fullständig implementation hade avståndsmätning med RSS föredragits då den är mest lättillgänglig.

Global translation, rotation och i vissa fall spegling av referenstriangeln är något som är oundvikligt för ankarlösa IPS-system och ingen begränsning av algoritmen i sig. Detta konstaterades ursprungligen av Priyantha et al [14]. Här menas att när ett IPS-system endast utgår ifrån avstånd är dessa problem oundvikliga och bör hanteras i algoritmen.

> "In contrast, anchor-free algorithms use local distance information to attempt to determine node coordinates when no nodes have preconfigured positions. Of course, any such coordinate system will not be unique and can be embedded into another global coordinate space in infinitely many ways, depending on global translation, rotation, and possibly flipping. This limitation is fundamental to the problem specification, and is not a limitation of the algorithm"[14, p.340]

De tre fallen var de mest utmanande problemen att lösa under arbetets gång, framförallt speglingen då denna endast kan avgöras via fasta positioner, exempelvis GPS-koordinater. Lösningen för speglingen som innebär att rutten itereras genom kräver att en virtuell referensriktning mot norr jämförs med den simulerade kompassen. Denna metod kan i vissa fall bli felaktig om riktningen mellan mätningarna uppskattas fel i förhållande till virtuella referensriktningen. Speglingen är alltså inte 100% trovärdig. Rotationen hade en lättare lösning då denna kan avgöras genom att använda 9DoF (s.8). Kompassen simuleras genom att bestämma en riktning från en mätpunkt till den nästa vilket gör att mätfel kommer troligtvis orsakas av 9DoF i verkligheten, eftersom mobilenheten exempelvis inte har ett stabilt läge. Global translation å andra sidan är inget som är möjligt för denna implementation såvida GPS inte inkluderas i någon av referenspunkterna för att kunna translatera inomhuspositioneringssystemet globalt. GPS-teknik var något som skulle uteslutas i syftet med arbetet.

WiFi-fingerprinting som användes av några av de arbeten som togs upp tidigare är en lämplig implementation för denna algoritm då denna lagrar signalstyrkan och koordinater i en databas. En nackdel är att vid ombyggnationer eller förflyttning av möbler så måste databasen ständigt uppdateras.

En kombination av Wi-Fi och Bluetooth visade bäst resultat av de arbeten som beskrevs tidigare vilket leder till konstaterandet att denna algoritm skulle

helst implementeras med en kombination av dessa två. Algoritmen har nämligen stöd för detta då den är oberoende av hårdvaran som används.

Anledningen till varför informationsstrukturerad omgivning togs upp var för att termen var intressant för IPS-systemens framtid. Orsaken till varför IPS inte används vardagligt än, trots den höga procentuella andelen som människor befinner sig inomhus, är att en optimerad standardlösning är obefintlig. Här spekulerades kring ifall ombyggnationer eller nya konstruktioner skulle satsa på en informationsstrukturerad omgivning, som att sätta sensorer under golvet, för att förenkla inomhuspositionering.

5.2 Diskussion av metod och resultat

Metoden som användes för att beräkna triangelns sidor har en noggrannhet som är beroende av hur rutten är utformad. Har användare gått mellan två beacons så har det rätta avståndet erhållits. Denna metod tar dock inte hänsyn till signalstörningar då avståndsmätningen kommer påverkas oavsett hur rutten ser ut i verkligheten. Trots denna nackdel så avser korrigeringen att överkomma denna problematik vilket visas att den gör i resultaten av testfallen som visar mindre felmarginal i korrigeringen.

Korrigeringen i sig fick två implementationer. Den ena använder Pythagoras sats och den andra cosinussatsen. Pythagoras sats har mer komplicerade ekvationer och ger dessutom inget bättre resultat än cosinussatsen vilket förklarar anledningen till varför denna valdes bort. Cosinussatsen visade sig vara fördelaktig både för att beräkningarna var enklare att implementera men också för att den visade tydligare varför problemet med att definiera vilken av beräningarna (addera eller subtrahera vinklarna) som skulle utföras uppstod. Ett annat sätt för att avgöra detta hade möjligen varit att använda en fjärde beacon. Genom att tillämpa så att algoritmen hanterar en fjärde beacon kan avstånden från mätpunkt till $b4$ användas för att bestämma var mätpunkten befinner sig och därmed avgöra vilken av uträkningarna som ska utföras. Detta är endast en spekulation och inget som verifierats teoretiskt.

Approximationen för att uppskatta det mest förekommande avståndet blev en slutgiltig lösning för att få fram ett resultat ur korrigeringen. Det hade varit önskvärt att vidareutveckla den ytterligare då den i dagsläget använder medelvärdet istället för minsta kvadratmetoden. Minsta kvadratmetoden är fördelaktig då den minimerar felet medan medelvärdet kan påverkas av felmätningar.

Implementationen i Java anpassade beräkningarna efter en 2-dimensionell värld men i verkligheten uppstår fler aspekter att ta hänsyn till, som att avståndsmätningen när mottagaren befinner sig precis under en beacon ger felaktigt resultat. För att algoritmen ska på ett optimalt sätt kunna användas i ett IPS-system bör den anpassas efter en 3D-miljö. En annan viktig vidareutveckling är att skapa en server som utför beräkningarna och en databas som lagrar mätdatat då en mobil enhet har för låg datakapacitet och batteritid.

Slutligen bör det poängteras att resultatet visade tydligt på att korrigeringen var värdefull för arbetets syfte då den validerade vägvalet som gjordes i metoden

när den emprisika studien avspeglade ett orimligt utförande inom projektets tidsramar.

Kapitel 6

Slutsatser

Inomhuspositionering är ett område som har hög potential att förbättra människors vardag med bättre orienteringsmöjligheter. Enligt litteraturstudien som gjordes har IPS möjlighet att bidra till många områden i samhället så som hälsa och social utveckling men det som hindrar tekniken är oförmågan att hantera signalstörningar på ett lämpligt sätt i inomhusmiljöer.

Algoritmen som skapades är generell i det avseendet att den är oberoende av hur avståndsmätningen mellan mottagare och sändare sker men även av nätverkets infrastruktur. Den kan klassas som en range-based anchor-free centraliserad algoritm där avståndsmätningen kan vara av distribuerad modell. Algoritmen kan tillämpas på många sätt och simuleringen kan användas för olika syften, exempelvis kan den underlätta installation av IPS-system.

Syftet med arbetet var att skapa en algoritm för ett ankarlöst IPS-system, helst med underlag för en empirisk undersökning. I tidigare forskning har implementationer av ankarlösa IPS-system använt nätverksinformationen, möjligtvis för att kompensera för de få ursprungspunkter som ankarlösa system kan utgå ifrån. Detta arbete har visat teoretiskt, via tillämpning av matematik, att nätverksinformationen tillsammans med noggrann avståndsmätning ej behövs. Däremot eftersom algoritmen inte implementerats empiriskt, vilket tidigare arbeten gjort, kan det vara möjligt att nätverksinformationen kan ge bättre avståndsmätning för ett Wi-Fi baserat IPS-system.

Ett stort vägval som gjordes under arbetets gång var förkastandet av den empiriska studien för att istället främja utvecklingen av korrigeringen. Valet visades sig i resultatet vara gynnsamt då testfallen påvisade att korrigeringen hade lägre felmarginaler vid tillägg av simulerade störningar.

Den befintliga algoritmen i dagsläget kan fortfarande inte hantera fler än tre beacons. Detta är en nödvändig utökning för att kunna implementera algoritmen i ett verkligt IPS-system. Korrigeringen bör även den förbättras genom att tillämpa minsta kvadratmetoden istället för medelvärdesberäkning vid approximationen. Felmarginaler i värdena som beräknats i algoritmen och som används i korrigeringen kan analyseras och bearbetas med avancerade statistiska metoder.

Arbetet har krävt ett brett datatekniskt kunnande då den täcker många områden som tillämpats under utbildningens gång. I större grad har matematik och programmering använts men även kunskaper i elektronik då arbetet även behandlade signaler.

Litteraturförteckning

[1] Arivan S. Bastos, Vaninha Vieira, and Antonio L. ApolinArio, Jr. Indoor location systems in emergency scenarios: A survey. In *Proceedings of the Annual Conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective - Volume 1*, SBSI 2015, pages 34:251–34:258, Porto Alegre, Brazil, Brazil, 2015. Brazilian Computer Society.

[2] Bing Hwa Cheng, Lieven Vandenberghe, and Kung Yao. Distributed algorithm for node localization in wireless ad-hoc networks. *ACM Trans. Sen. Netw.*, 6(1):8:1–8:20, January 2010.

[3] Kevin Curran, Eoghan Furey, Tom Lunney, Jose Santos, Derek Woods, and Aiden McCaughey. An evaluation of indoor location determination technologies. *Journal of Location Based Services*, 5(2):61–78, 2011.

[4] Carlos E. Galván-Tejada, José C. Carrasco-Jiménez, and Ramon F. Brena. Bluetooth-wifi based combined positioning algorithm, implementation and experimental evaluation. *Procedia Technology*, 7:37–45, 2013.

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[6] Mohammad Reza Gholami. *Positioning algorithms for wireless sensor networks*. Institutionen för signaler och system, Kommunikationssystem, Chalmers tekniska högskola,, 2011. 56.

[7] Marco Gruteser. Indoor localization: Ready for primetime? In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, pages 375–376, New York, NY, USA, 2013. ACM.

[8] Naveed Ul Hassan, Aqsa Naeem, Muhammad Adeel Pasha, Tariq Jadoon, and Chau Yuen. Indoor positioning using visible led lights: A survey. *ACM Comput. Surv.*, 48(2):20:1–20:32, November 2015.

[9] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks.

In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, pages 81–95, New York, NY, USA, 2003. ACM.

[10] Amin Karbasi. From centralized to distributed sensor localization. In *Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students*, S3 '10, pages 5–8, New York, NY, USA, 2010. ACM.

[11] Dae-Kyung Kim, Yeong-Sam Kim, and Jong-Wha Chong. Hybrid rss/toa wireless positioning with a mobile anchor in wireless sensor networks. In *Proceedings of the 6th Latin America Networking Conference*, LANC '11, pages 25–32, New York, NY, USA, 2011. ACM.

[12] Nisarg Kothari, Balajee Kannan, Evan D. Glasgwow, and M. Bernardine Dias. Robust indoor localization on a commercial smart phone. *Procedia Computer Science*, 10:1114 – 1120, 2012. {ANT} 2012 and MobiWIS 2012.

[13] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition, 2008.

[14] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. Poster abstract: Anchor-free distributed localization in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 340–341, New York, NY, USA, 2003. ACM.

[15] Yoonseok Pyo, Kouhei Nakashima, Shunya Kuwahata, Ryo Kurazume, Tokuo Tsuji, Ken'ichi Morooka, and Tsutomu Hasegawa. Service robot system with an informationally structured environment. *Robotics and Autonomous Systems*, 74, Part A:148 – 165, 2015.

[16] Prasan Kumar Sahoo, I-Shyan Hwang, and Shi-Yao Lin. A distributed localization scheme for wireless sensor networks. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 77:1–77:7, New York, NY, USA, 2008. ACM.

[17] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '03, pages 201–212, New York, NY, USA, 2003. ACM.

[18] Stefan Steiniger, Moritz Neun, and Alistair Edwardes. Foundations of location based services. Cartography 1:1–11, 2016.

[19] Guang Tan, Hongbo Jiang, Shengkai Zhang, Zhimeng Yin, and Anne-Marie Kermarrec. Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks. *ACM Trans. Sen. Netw.*, 10(1):6:1–6:21, December 2013.

[20] Wasiq Waqar, Yuanzhu Chen, and Andrew Vardy. Incorporating user motion information for indoor smartphone positioning in sparse wi-fi environments. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, pages 267–274, New York, NY, USA, 2014. ACM.

Selma Abbassi, ndi13sai@student.hig.se
Rickard Engström, nbt09rem@student.hig.se

Handledare: Åke Wallin, ake.wallin@hig.se
Examinator: Jonas Boustedt, jonas.boustedt@hig.se

Syntronic: Håkan Sjörling, hast@syntronic.se
Magnus Sandberg, masa@syntronic.se

Kapitel 7

Appendix

7.1 Metod

7.1.1 Arbetsmodeller och processverktyg

Scrum

Scrum är en agil utvecklingsmetod för systemutveckling skapad av Jeff Sutherland och Ken Schwaber. Arbetsformen innebär ett flexibelt tillvägagångssätt där projektet delas upp i sprintar. Varje sprint kan utsträcka sig över 1-2 veckor och denna inleds med ett sprintmöte där medlemmar diskuterar vad som ska åstadkommas samt evaluerar resultat från tidigare utförda uppgifter. Metoden baseras på idén att fasta planeringar leder till oförmåga att anpassa sig för oförväntade problem.

The Agile Manifesto

Det agila manifestet bygger på fyra principer som ämnar att förbättra systemutveckling.

1. Individer och interaktioner framför processer och verktyg. Att ständigt ha kontakt med sina medarbetare eller sin kund minskar chansen för missförstånd. Trots att utvecklingsprocessen och verktygen som används är viktiga att lägga fokus på så bör interaktionen med de inblandade individerna prioriteras. I detta arbete har författarna närvarat på arbetsplatsen dagligen och utvecklat programvaran tillsammans. Detta leder till en snabbare fortskridande process och ett bättre resultat.

2. Fungerande programvara framför omfattande dokumentation. Arbetet har planerats veckovis och även dagliga planeringar har gjorts. Detta har dokumenterats ständigt men i första hand har programvaran kontrollerats att den är fungerande.

3. Kundsamarbete framför kontraktsförhandling. Samarbetet med Syntronic har inneburit ständiga uppdateringar om arbetet för kunden. Den dagliga närvaron på arbetsplatsen med lättåtkomlig kund har ökat möjligheten att kunna diskutera och anpassa arbetet efter behov.

4. Anpassning till förändring framför att följa en plan. Ingen detaljerad och generell plan skapades i början av arbetet utan istället har det bestämts arbetsuppgifter för varje vecka på måndagar. Detta ger utrymme för förändring.

TDD: Test Driven Development

Testdriven utveckling är en systemutvecklingsmetod där automatiserade enhetstester skapas för väsentliga uträkningar i algoritmen. Testerna ska skapas innan man börjar skriva produktionskod. Testet bör först fallera och man ska endast skriva tillräckligt med produktionskod för att testet ska gå igenom.

Meningen med detta är att produktionskoden och testerna ska gå hand i hand. Med enhetstester tillåts programmet vara öppet för modifiering och utökning.

Verktyg

JavaFX, Github, Trello, GeoGebra, Android Studio, IntelliJ.

7.1.2 Systemdesign

Refaktorisering av matriser

Refaktorisering av matriser visas i exempelkoden nedan.

```
public class Matrix22 {

    protected Position v1, v2;

    public Matrix22(Position v1, Position v2) {
        this.v1 = new Position(v1.getX(), v1.getY(), 0);
        this.v2 = new Position(v2.getX(), v2.getY(), 0);

    }

    public Position multiply(Position position)
    {
        double x = v1.getX() * position.getX()
                        + v1.getY() * position.getY();
        double y = v2.getX() * position.getX()
                        + v2.getY() * position.getY();

        return new Position(x, y, position.getZ());
    }
```

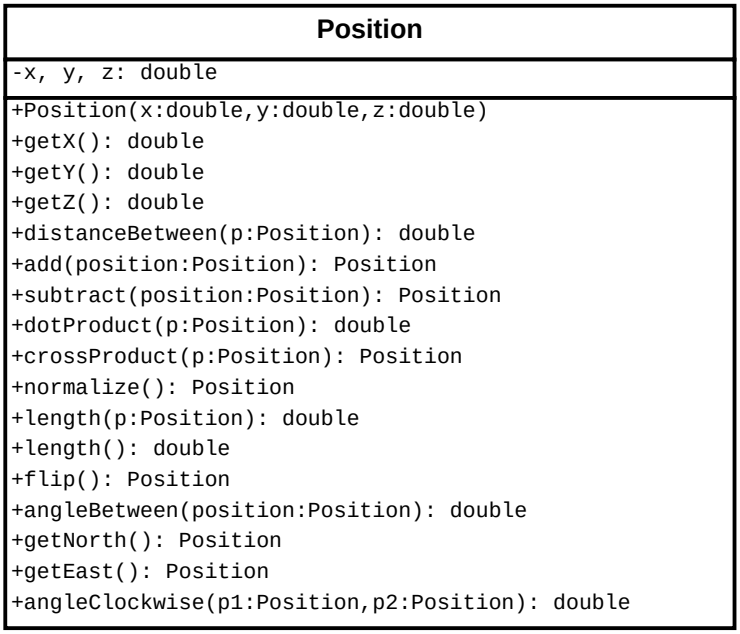
```
}
```

```
public class ClockwiseRotationMatrix extends Matrix22{


    public ClockwiseRotationMatrix(double angleRadians) {
        super(new Position(Math.cos(-angleRadians),
                           -Math.sin(-angleRadians), 0),
                    new Position(Math.sin(-angleRadians),
                            Math.cos(-angleRadians), 0));
    }
```

Immutable

Exempel på en immutable-klass `Position`



Figur 7.1: Klassen `Position`

S.O.L.I.D principer

Tabell 7.1: S.O.L.I.D principer

S	SRP	Single Responsibility Principle	En klass ska ha endast ett ansvar. Endast en potentiell ändring i programvarans specifikation ska kunna påverka specifikationen i klassen.
O	OCP	Open/closed Principle	Mjukvaruentiteter ska vara öppna för vidareutveckling men stängda för modifikation.
L	LSP	Liskov Substitution Principle	Objekt i ett program ska vara utbytbara med instanser av deras subtyper utan att alterera trovärdigheten för det programmet.
I	ISP	Interface Segregation Principle	Flera klient-specifika interface är bättre än ett generellt interface med gemensamma bestämmelser.
D	DIP	Dependency Inversion Principle	Man ska vara beroende av abstraktioner och inte konkretioner.