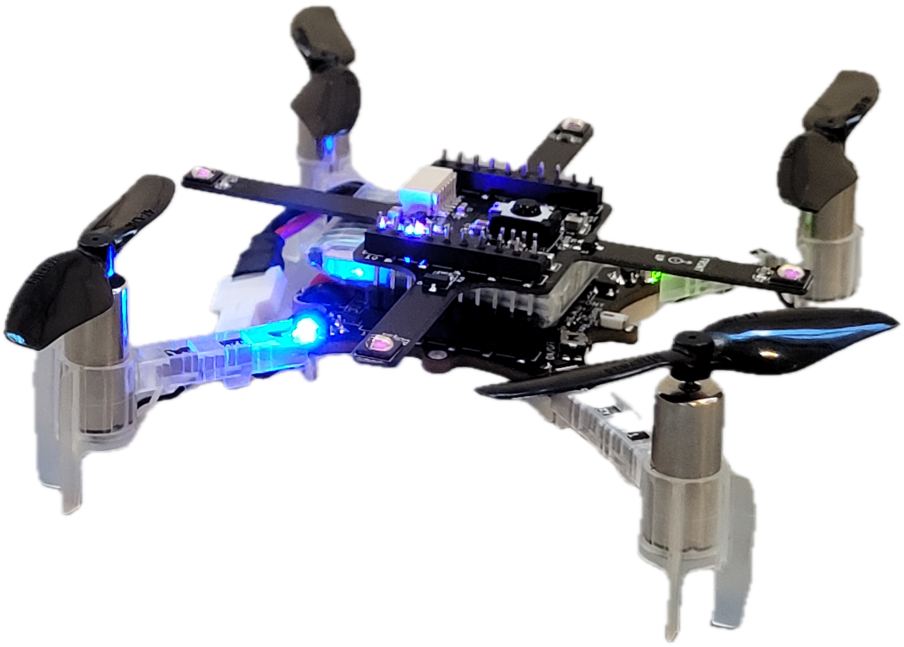
CHALMERS



Koordinering av drönare för autonom områdesavsökning

Hassan Ahmad, Alma Edling, Oscar Larsson,
Ninus Sellin Bredberg, Jesper Skoglund

INSTITUTIONEN FÖR ELEKTROTEKNIK

Chalmers Tekniska Högskola

Göteborg 2026

www.chalmers.se

Kandidatarbete 2026

Koordinering av drönare för autonom områdesavsökning

Hassan Ahmad
Alma Edling
Oscar Larsson
Ninus Selling Bredberg
Jesper Skoglund

CHALMERS

Institutionen för Elektroteknik
Chalmers Tekniska Högskola
Göteborg 2026

Koordinering av drönare för autonom områdesavsökning

Hassan Ahmad
Alma Edling
Oscar Larsson
Ninus Sellin Bredberg
Jesper Skoglund

Handledare: Knut Åkesson, Institutionen för elektroteknik
Examinator: Martin Fabian, Institutionen för elektroteknik

Kandidatarbete 2026
Institutionen för elektroteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Närbild på en Crazyflie-drönare med Active marker deck monterat.

Skriven i LaTeX
Göteborg 2026

Abstract

This study investigates how drones of the model Crazyflie2.1+, in combination with a motion capture system, can be used for autonomous area scanning and, upon detection, tracking and streaming of the positional data of a moving target.

To enable a drone system capable of autonomously switching between area scanning and target tracking, a control script was developed utilizing Qualisys track manager and Crazyswarm2.

The results show that the drones were able to follow a predefined search trajectory and, upon detection of a moving target, transition to target tracking while constantly streaming its position to an external system. However, the transition to area scanning after tracking was not implemented due to time constraints.

This study demonstrates the potential of motion capture-based drone systems for autonomous navigation and target tracking in controlled environments, while also highlighting challenges related to the implementation of such systems.

Sammanfattning

Denna studie undersöker hur ett drönarsystem med drönare av modellen Crazyflie 2.1+ i kombination med ett motion capture-system kan användas för autonom avsökning av ett område samt, vid detektion, efterföljning samt vidarebefordran av ett rörligt måls positionsdata.

För att möjliggöra ett drönarsystem som autonomt kan byta mellan områdesavsökning samt efterföljning utvecklades ett styrskript som använder sig av Qualisys Track Manager samt Crazyswarm2.

Resultatet visar att drönarna kunde flyga enligt en avsökningsbana samt, vid detektion av det rörliga målet, övergå till efterföljning samt vidarebefordra dess positionsdata. På grund av tidsbrist kunde tillståndsbytet mellan avsökning och efterföljning inte implementeras.

Studien visar potentialen hos motion capture-baserade drönarsystem för autonom navigering och efterföljning av mål i kontrollerade miljöer, samtidigt som den belyser utmaningar kopplade till att förverkliga ett sådant system.

Tillkännagivande

Vi vill inleda denna rapport med att rikta ett stort tack till vår handledare Knut Åkesson, samt Rikard Karlsson, Kilian Tamino Freitag, Kristian Ceder och Gunnar Edman för deras stöd och vägledning under projektets gång. Ett särskilt stort tack riktas även till Axel Steffner som var medverkande i projektets tidiga stadie. De insatser Axel bidrog med var ovärderliga för vårt arbete genom hela projektets gång.

Hassan Ahmad, Alma Edling, Oscar Larsson,
Ninus Sellin Bredberg, Jesper Skoglund

Göteborg, Maj 2026

Akronymer

Akronymer som används i rapporten:

6DoF	Six degrees of freedom
CSV	Comma Separated Value
GPS	Global Positioning System
MoCap	Motion Capture
PWM	Pulse Width Modulation
QTM	Qualisys Track Manager
ROS 2	Robot Operating System 2
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol

Innehåll

1

Introduktion

Koordinerade autonoma drönare utgör ett växande forskningsområde inom utvecklingen av obemannade flygsystem. Denna rapport presenterar och utvärderar ett drönarsystem där flera drönare samarbetar för att avsöka ett område efter ett rörligt mål. Då en drönare upptäcker målet övergår drönaren till efterföljning och vidarebefordring av målets position till ett externt system. Drönarnas styrdirektiv förmedlas genom en centraldator som kommunicerar med samtliga aktiva drönare i området.

1.1 Bakgrund

Obemannade luftfarkoster (UAV) har under de senaste decennierna genomgått en omfattande expansion avseende användningsområden, från specialiserade militära uppdrag till breda civila tillämpningar. Ett exempel återfinns inom jordbrukssektorn [1], där tekniska framsteg inom 5G och artificiell intelligens har möjliggjort implementation av autonoma UAV-system i alltmer komplexa miljöer. Utvecklingen har även drivits av förbättrad batteriteknik och reducerade produktionskostnader, vilket har möjliggjort nya tillämpningsområden för UAV:er [2].

Den förbättrade prestandan och autonomin hos UAV:er har öppnat för nya möjligheter gällande UAV-koordinering. Genom att agera som en koordinerad enhet kan fördelar såsom effektivitet, skalbarhet och robusthet genom redundans erhållas. Denna potential är särskilt tydlig i tidskritiska räddningsinsatser [3], där det framgår att flertalet UAV:er bidrar till effektivare avsökning med robust objektidentifiering.

För att realisera samverkan av UAV:er krävs en underliggande styrarkitektur, där en grundläggande distinktion görs mellan centraliserade och decentraliserade lösningar [4]. Dessa lösningar avgör hur det autonoma systemet implementeras och fungerar i praktiken.

I en centraliserad styrarkitektur ansvarar en centralenhet, exempelvis en centraldator, för att kommunicera med samtliga UAV:er. Detta sker främst genom datainsamling och bearbetning för att därefter sända lämpliga styrkommandon. I decentraliserade styrarkitekturer sker beslutsfattandet istället lokalt hos varje UAV. Detta möjliggörs av interna sensorer och ett kommunikationsnätverk för informationsutbyte mellan UAV:erna.

För detta projekt valdes drönare (4-rotoriga UAV:er) av modellen Crazyflie 2.1+ som tillverkas av det svenska företaget Bitcraze AB. Dessa valdes eftersom de lämpar sig väl för utveckling, testning samt utvärdering av olika projekt där drönare är centrala.

Bitcraze AB har ett samarbete med det svenska företaget Qualisys AB som tillverkar motion capture-system (MoCap). Detta innebär att särskilda markörer har tillverkats för drönarna som upptäckas av MoCap-systemet vilket tillät mycket precis och kontinuerlig datainsamling.

Ett sådant MoCap-system har köpts in och monterats på Chalmers tekniska högskola, där även ovannämnda drönare finns tillgängliga. Med detta som bakgrund fanns det en möjlighet att realisera och utvärdera ett koordinerat drönarsystem.

1.2 Problem

Implementering av koordinerade drönarsystem kräver integration mellan flera hårdvaru- och mjukvaruplattformer med varierande kommunikations-, styr- och positionssystem. Detta medför utmaningar kopplade till systemintegration, realtidskommunikation och samverkan mellan distribuerade komponenter. Projektet behandlar hur ett sådant system kan realiseras.

1.3 Syfte

Syftet med detta projekt är att, baserat på befintlig forskning, implementera ett system för koordinering av drönare som samverkar för att utföra flera uppgifter. Dessa består av att söka efter ett rörligt mål inom ett avgränsat område samt, vid detektion av målet, övervaka och vidarebefordra målets position till ett externt system vars syfte är att omringa målet.

1.4 Avgränsningar

Följande avgränsningar gjordes för att anpassa projektets omfattning till den tillgängliga tidsramen och en rimlig komplexitet för ett kandidatarbete.

- Detaljerna i det externa systemet, vars syfte är att omringa målet, behandlades ej.
- Implementerade algoritmer optimerades ej.
- Använd hårdvara modifierades ej.
- Medföljande hårdvaruspecifik mjukvara modifierades ej.
- Avsökningsområdet begränsades till ett rektangulärt område i eventhallen på Chalmers tekniska högskola.

2

Ansats

I detta kapitel presenteras samtliga delproblem vars lösningar tillsammans ska realisera ett system som uppnår projektets syfte. Potentiella lösningar på varje delproblem presenteras kortfattat följt av den valda lösningen och motivering för denna lösning. Avslutningsvis presenteras hårdvaru- och mjukvarukomponenter samt deras roll i den slutgiltiga implementationen.

2.1 Positionering

Koordinering av flera drönare ställer höga krav på noggrann och tillförlitlig positionering, vilket innefattar både inhämtning av positionsdata samt förmåga att dirigera drönarna. Positionsinhämtning kan realiseras med olika metoder, exempelvis GPS, interna sensorer som accelerometrar och gyroskop, eller MoCap-system. För detta projekt valdes ett externt MoCap-system eftersom det uppfyller krav på hög noggrannhet samt möjligheten till högfrekvent realtidsuppdatering.

Drönarpositionering kan realiseras med olika nivåer av abstraktion. På låg abstraktionsnivå kan styrning ske genom direkt kontroll av motorernas effekt, medan högre abstraktionsnivåer möjliggör styrning via förflyttningskommandon där intern reglering hanterar drönarens underliggande dynamik. Inom detta projekt användes diverse förflyttningsdirektiv som hårdvaran möjliggör, där direktiven både hade olika nivåer av abstraktion, detta för att utnyttja dess respektive fördelar.

2.2 Avsökning

Utöver problematiken med positioneringen utgör även avsökning ett delproblem inom projektet. Problematiken vid utformningen av avsökningsalgoritmer för drönarsvärmar är optimeringen av täckningsgraden i förhållande till tidsåtgången. För att maximera varje enskild drönares bidrag till sökinsatsen tilldelas varje drönare varsitt icke överlappande delområde.

För att öka arean av varje enskild drönares täckningsyta är batteritid en viktig faktor. Eftersom mjuka rörelser belastar drönarens batteri mindre än ryckiga rörelser [5] skapades en banfilsgenerator som skapar filer i ett format som möjliggör komplexa avsökningsmönster med mjuka rörelser.

2.3 Detektion av inkräktaren

Det sista delproblemet för att lösa projektets helhet är att kunna detektera det rörliga målet, hädanefter kallat inkräktaren. Drönare är ofta utrustade med kameror som i kombination med intern bildbehandling möjliggör detektion och klassificering av objekt. Eftersom drönarna som användes i projektet inte var utrustade med kameror simulerades artificiella synfält baserade på geometrisk modellering och positionsdata från MoCap-systemet. Detektionen baserades på att avgöra om inkräktaren befann sig inom de simulerade synfälten.

2.4 Projektets hårdvara

MoCap-systemet, drönarna samt Active marker deck utgör de centrala hårdvarukomponenter som användes i projektet. Dessa komponenter valdes för att dom är kompatibla med varandra.

2.4.1 Motion Capture-systemet

För att möjliggöra stabil styrning och navigering krävs tillförlitlig och noggrann positionsdata för drönarna. MoCap-systemet som användes i projektet består av åtta Qualisys A12 Arqus-kameror, se Figur 2.1. Systemet har förmågan att förmedla information om drönarnas position och orientering med en frekvens om 300 Hz med en felmarginal mindre än 0.04 mm[1]. Vidarebefordringar av informationen skedde genom en UDP-server (eng. *User Datagram Protocol*) som kamerornas medföljande mjukvara Qualisys Track Manager (QTM) tillhandahåller.

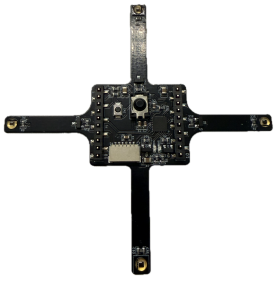
[1] https://docs.qualisys.com/qtm/content/techref/qualisys_sensor_specs_marker.htm, hämtad 7/5/2026

Figur 2.1: Bilden visar eventhallen där Mocap-systemet befinner sig, fem av systemets totalt åtta kameror är inringade i rött.

2.4.2 Drönarna

Drönarna som användes i projektet är fyr-rotor UAV:er av modellen Crazyflie 2.1+ [2], se Figur 2.2. För positionering med hjälp av MoCap-systemet användes tilläggskortet Active marker deck[3], se Figur 2.3. Detta tilläggskort tillhandahåller funktionalitet för att MoCap-systemet ska kunna följa drönarens position samt orientering i rummet. Tillsammans med MoCap-systemet löser detta delproblemet med noggrann positionsinhämtning.

Figur 2.2: Drönare med Active marker deck monterat.



Figur 2.3: Active marker deck.

[2] https://www.bitcraze.io/products/crazyflie-2-1-plus/ hämtad 7/5/2026

[3] https://www.bitcraze.io/products/active-marker-deck/ hämtad 7/5/2026

2.5 Projektets mjukvara

Projektet använde flera olika mjukvaror, Robot Operating System 2 (ROS2) samt Crazyswarm2, som tillsammans kördes inuti en Docker-container. Dessa del-system kommunicerar konstant med varandra och tillsammans koordinerar de hela drönarsystemet. Docker-containern uppgift är främst att underlätta driftsättningen av hela projektet.

2.5.1 Robot Operating System 2

Eftersom projektet har flera olika mjukvaru- och hårdvarukomponenter som i realtid måste kommunicera med varandra är ett effektivt och koordinerat system som möjliggör detta nödvändigt. Detta projekt använde därför mjukvaran ROS2 [6]. ROS 2 tillhandahåller funktionaliteten att skapa varje delprocess som en ROS2-nod. Dessa noder kan sedan lyssna på (eng. *subscribe*) och publicera (eng. *publish*) information på ämnen (eng. *topics*), vilket möjliggör utbyte av information mellan systemets alla noder.

2.5.2 Crazyswarm2

Crazyswarm2 är en portering av det ursprungliga Crazyswarm till ROS 2[7]. Denna används för kommunikation mellan centraldatorn, MoCap-systemet samt drönarna. Crazyswarm2-delprocessens ROS2-nod hanterar dirigering, handskakning med, samt vidarebefordran av positionsdata till drönarna genom att kommunicera med Crazyradio[4]. Crazyswarm2 tillhandahåller även Python-paket som möjliggör extern dirigering via egenskriven kod. Därtill finns även en simulator där test kan utföras innan de körs i verkligheten. Denna simulator användes för testning av nya implementeringar.

2.5.3 Exekveringsmiljö

Docker är en containerbaserad plattform som möjliggör paketering av applikationer och deras externa paketberoenden i en isolerad miljö. Detta möjliggör att drönarsystemet kan användas oavsett vilken typ av dator som kör systemet. Docker underlättar även hantering och uppdatering av de externa paket som krävs för att upprätthålla drönarsystemets funktionalitet[5]. Eftersom drönarsystemet består av olika komponenter som kräver specifika externa paket användes Docker för modulariteten som det erbjuder.

[4] https://www.bitcraze.io/products/crazyradio-2-0/, hämtad 13/5/2026

[5] https://docs.docker.com/?utm_source=docker&utm_medium=inproductad&utm_campaign=20-11nurturecli_docs, hämtad 7/5/2026

3

Drönarpositionering

Detta kapitel behandlar hur adekvat positioneringsdata kan erhållas genom ett MoCap-system samt dess bakomliggande teori. Därefter presenteras de olika förflyttningsdirektiv som finns tillgängliga för drönarplattformen följt av en genomgång av systemets inställningar samt aktuell tillämpning av drönarens olika förflyttningsdirektiv. Avslutningsvis diskuteras systemets prestanda samt potentiella förbättringar.

3.1 Positionsestimering med Motion capture

Positionsestimering med hjälp av ett MoCap-system fungerar genom att triangulera punkter på ett objekt vars position ska estimeras. Dessa punktmarkörer kan antingen vara aktiva eller passiva.

Passiva markörer reflekterar infrarött ljus genom dess reflektiva ytskikt där kamerorna både skickar ut och därefter detekterar detta infraröda ljus. Dessa markörer kan göras väldigt lätta och i olika storlekar [8].

Aktiva markörer skickar ut eget infrarött-ljus som kamerorna detekterar. Aktiva markörer är därför bättre än passiva om MoCap-området har många reflektiva ytor som kan störa systemet. En annan fördel är att de använder sig av sekventiell kodning för automatisk igenkänning av kroppar [9]. Däremot kräver aktiva markörer elektroniska komponenter, vilket gör dom dyrare än passiva markörer.

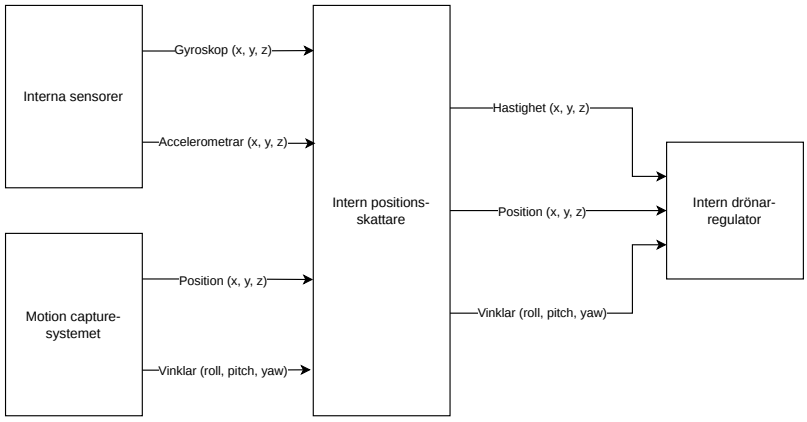
Minst två kameror måste samtidigt se samtliga aktiva eller passiva markörer som kroppen består av för att kunna estimera dess position. Om det är en kropp med sex frihetsgrader (6DoF) kan kamerorna även estimera dess rotationsmatris. Däremot är det önskvärt att så många kameror som möjligt ser alla markörer då detta minskar felmarginalen eftersom mer data finns tillgängligt för trianguleringsberäkningarna.

MoCap-systemet användes för att erhålla en bättre estimering av drönarens x,y,z position samt rotationsmatris än vad som kan erhållas från drönarens interna sensorer. Systemets förmåga att högfrekvent skicka noggrann positioneringsdata möjliggör mer avancerad rörelselogik då felmarginalen på positionsestimeringen är låg.

3.2 Drönarens positionsestimering

För att möjliggöra stabil flygning samt styrning av drönare används olika system som omfattar extern positionsestimering, intern reglering samt extern dirigering, där önskad förflyttning är kontrollerat av olika styrkommandon. Denna struktur möjliggör en uppdelning mellan reglering och styrning samt möjligheten att ha olika nivåer av abstraktion.

Crazyflie-drönarnas plattform är uppbyggd kring en intern struktur för tillståndsskattning och reglering där sensordata, såsom position och orientering, först behandlas av en tillståndsskattare, se Figur 3.1. Denna skattning kombinerar mätdata från tillgängliga sensorer med en modell av systemets dynamik för en mer robust uppskattning av drönarens tillstånd. Positionsinformationen tillhandahålls av MoCap-systemet, vilket integreras i skattningsprocessen och kompletterar de interna sensorerna. Den estimerade positionen och orienteringen används därefter som indata till drönarens interna regulator. Regulatorn ansvarar därefter för att stabilisera samt styra drönaren.



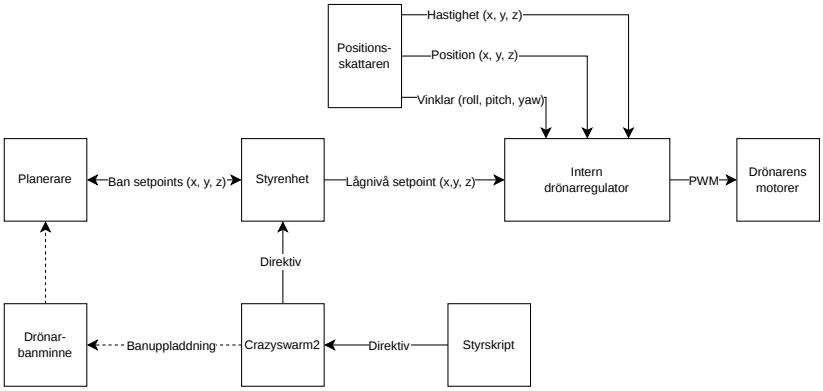
Figur 3.1: Flödesschema över drönarens interna skattare för tillståndsestimering

3.3 Drönarens förflyttningsarkitektur

Förflyttning av drönarna kan ske på olika abstraktionsnivåer, högnivåkommandon respektive strömning av lågnivå-setpoints [10]. En setpoint är en punkt i rummet dit man vill att drönaren ska förflytta sig till. Punkten måste vara tillräckligt nära drönarens nuvarande position, annars kommer regulatorn generera orimliga accelerationer för att genomföra förflyttningen, vilket resulterar i ostabil flygning.

All förflyttning realiseras genom kontinuerlig strömning av lågnivå-setpoints till drönaren. Vid användning av högnivåkommandon genererar styrenheten automatiskt en sekvens av lågnivå-setpoints som motsvarar den önskade rörelsen, vilka därefter kontinuerligt strömmas till drönaren. Detta till skillnad från manuell strömning av lågnivå-setpoints, där sekvensen som uppfyller önskad rörelse måste konstrueras på egen hand.

Vid mottagandet av lågnivå setpoints beräknar drönarens interna regulator ut motorstyrsignalerna som kommer generera lämplig förflyttning. Där motorstyrningen sker genom pulsbreddsmodulering (PWM) av motorernas ström. Se Figur 3.2 för ett flödesschema av styrdirektiv samt hur realisering av drönarförflyttning sker.



Figur 3.2: Hur förflyttningsdirektiv behandlas av olika moduler samt hur önskad rörelse realiseras av drönaren.

3.3.1 Högnivåkommandon

Tillgängliga högnivåkommandon är exempelvis *goTo*, *startTrajectory*, *takeoff* och *land*. Dessa kommandon ger hög användarvänlighet då stora delar av den bakomliggande processen ej görs av användaren, med nackdelen att förmågan att finjustera rörelsen begränsas.

3.3.2 Strömning av lågnivå setpoints

Strömmning av lågnivå setpoints sker genom *cmdPosition* samt *cmdFullstate*. Strömmning av lågnivå setpoints har en lägre grad av abstraktion och kräver därmed större insikt i systemets dynamik samt dess begränsningar. I gengäld ger det större kontroll och möjlighet till finjustering av drönarens rörelse.

Kommandot *cmdFullstate* ger dig möjlighet att strömma en lågnivå-setpoint samt att styra hastighet, acceleration, yaw och vinkelhastighet som önskas appliceras för att nå den. Där *cmdPosition* strömmar en lågnivå-setpoint, men drönarens interna styrenhet räknar ut hastighet, acceleration samt vinkelhastighet, men användaren fortfarande kan specificera önskad yaw.

3.4 Konfiguration av Mocap-systemet

Mocap-systemet konfigurerades i dess medföljande mjukvara QTM där rigida kroppar med 6DoF skapades. En rigid kropp definierades genom de 4 punkterna som Bitcraze Active marker deck skickar ut i rummet.

Vid konfigurering av QTM-projektet användes en exponeringstid på 450 μs för samtliga kameror, detta i enlighet med Qualisys egna rekommenderade inställningar[1]. Efter att de rigida kropparna definierats och verifierats i QTM kontrollerades att informationen skickades samt var korrekt genom att lyssna på vilken data MoCap-systemets ROS2-nod publicerade.

Detta kontrollsteg upprepades innan något förflyttningsscript testades. Detta eftersom drönarna fortfarande försöker flyga och utföra sin uppgift trots avsaknad av information från MoCap-systemet, vilket leder till att dom flyger instabilt och kraschar.

3.5 Användning av förflyttningsdirektiv

Förflyttning av drönarna skedde främst genom en kombination av lågnivå och högnivå-kommandon efter önskad abstraktionsnivå, då kombinationen möjliggör mer avancerade manövrar och koordinering.

[1] https://www.qualisys.com/video-tutorials/how-to-fly-a-crazyflie-quadcopter-with-qualisys/, hämtad 7/5/2026

3.5.1 Högnivå kommandon

Högnivåkommandon som användes inkluderar *Takeoff*, där parametrarna utgjordes av önskad höjd och utförelsetid. Kommandot användes för att initialt lyfta drönaren till önskad höjd. På samma kommando-nivå finns även *Land*, med motsvarande parametrar, för att sänka drönaren till en viss höjd och därefter stänga av motorerna.

För diverse olika förflyttningar i rummet användes *goTo*, där önskad slutposition i rummet (x, y, z), *yaw* samt utförelsetid specificerades. Detta möjliggjorde styrning av drönaren till specifika punkter.

Vid mer komplexa mjuka rörelser användes *startTrajectory*, som utgick från antingen den nuvarande positionen för drönaren eller startpositionen, för att sedan flyga enligt den specificerade banan. Parametrar som baklängeskörning samt tidsfaktor för långsammare eller snabbare banflygning specificerades enligt önskat beteende.

3.5.2 Strömning av lågnivå-setpoints

Efterföljning av inkräktaren realiserades genom att konstant strömma lågnivå-setpoints genom kommandot *cmdPosition*, där önskad position (x, y, z) och *yaw* specificerades. Detta kommando användes för att det möjliggjorde effektiv positionsföljning och snabb respons vid ändringar i inkräktarens position.

Slutligen användes *notifySetPointStop*, med enbart tidsfördröjning som parameter. Detta kommando användes för att växla mellan högnivåkommandon samt strömning av lågnivå setpoints efter önskad tidsfördröjning och därmed styra vilken typ av kommando som drönaren var beredd på att ta emot.

3.6 MoCap-systemets prestanda och pålitlighet

MoCap-systemet fungerade väl med adekvat vidarebefordring av positionsdata för att erhålla stabil flygning vid såväl simpla som komplexa manövrar. Däremot var det under en del av projektets gång problem med markörerna. Detta eftersom QTM inställningarna samt hårdvaran enbart var konfigurerade för aktiva markörer under en majoritet av projektets gång.

När arbetet med detektering och efterföljning av inkräktaren påbörjades och därmed introduktionen av passiva markörer utöver de aktiva markörerna, började spökpunkter att dyka upp i rummet. Detta berodde på reflektiva ytor inom MoCap-området.

Problemet åtgärdades genom att montera aktiva markörer på den inkräktande RC-bilen eftersom Qualisys ej rekommenderar att använda både aktiva och passiva markörer i samma projekt[2].

Utöver problemen med den initiala blandningen av aktiva och passiva markörer var det även stundvis problem med att MoCap-systemet tappade bort drönarna i vissa rotationsorienteringar. Detta berodde på att några kameror inte var optimalt vinklade och därför syntes punkterna inte alltid av minst 3 kameror. Detta åtgärdades genom att krympa det område som drönarna fick befinna sig inom. En mer optimal lösning hade varit att vinkla om kamerorna eller köpa in fler kameror, alternativt både och.

3.7 Utvärdering av förflyttningsdirektiv

En majoritet av förflyttningsdirektiven fungerade på ett pålitligt sätt. Dessa inkluderar *Takeoff*, *Land* samt *startTrajectory*. Däremot var det återkommande fel med både *goTo* samt *notifySetpointsStop*.

Ett återkommande problem med *goTo*-kommandot var att drönarens positionsfel konsekvent låg i intervallet 15–30 cm. Felet definierades som det euklidiska avståndet mellan uppmätt position och specificerad målposition i tre dimensioner, baserat på positionsdata från MoCap-systemet. Orsaken till detta är inte helt klarlagd. Det är dock känt att drönaren betraktar en förflyttning som slutförd när den upphör att röra sig i x-, y- och z-led och istället hovrar på plats. Vid kontinuerlig loggning av drönarens aktuella position i förhållande till den beordrade målpositionen observerades att hovring initierades trots att drönaren fortfarande befann sig på ett betydande avstånd från målet. Avvikelsen var inte konsekvent i varken x- eller y-led, däremot var noggrannheten i z-led generellt högre. Samtidigt verifierades att positionsdatan från MoCap-systemet var korrekt, vilket indikerar att felet sannolikt inte beror på MoCap-systemets data, utan från styrningen eller tolkningen av dess positionsdata.

Kommandot *notifySetpointsStop*, fungerade inte som beskrivet. Detta trots att användningsintruktionerna följdes noggrant samt olika variationer av dess användning testades. Vad detta beror på är okänt och kunde inte undersökas vidare på grund av tidsbegränsningar.

Det bör dock noteras att versioner av mjukvaran som användes i detta arbete kan va-

[2] https://docs.qualisys.com/qtm/content/running_the_system/qualisys_active_markers.htm#How2, hämtad 7/5/2026

ra utfasad vid framtida läsning. Detta kan innebära att funktionaliteten förbättrats eller åtgärdats i uppdaterade versioner.

4

Avsökning

Detta kapitel redogör för de teoretiska modeller bakom valda avsökningsmönster samt den tekniska processen för att transformera dessa till exekverbara banfiler. Fokus ligger på att förena geometrisk täckning med drönarnas fysiska begränsningar.

4.1 Avsökningsmönster

Följande avsnitt förklarar den teori samt de matematiska formler och begrepp som ligger till grund för generering av de banfiler som används för drönarnas avsökningsmönster.

4.1.1 Spiralsökning

Vid sökning utifrån en känd punkt, där det dynamiska målet redan har detekterats, finns flera etablerade mönster som *Expanding square search*, *Sector search* och spiralsökning [11, s. 23, avsnitt 3]. *Expanding square search* är en etablerad standard, men sökmönstret omfattar rätvinkliga svängningar vilket tidigare diskuterats (se avsnitt 2.2, speciellt [5]) vara önskvärt vid framförande av UAV-system. *Sector search* beskrivs som ett vanligare alternativ för begränsade sökområden med krav på noggrannhet, metoden lämpar sig dock mindre bra för större avsökningsområden som kräver effektiv sökning.

Vidare beskrivs spiralsökning som är nära besläktad med *Expanding square search*, spiralsökning undviker skarpa svängar och lämpar sig väl då man vill hålla accelerationen mjuk. Med detta i åtanke valdes spiralsökning som en avsökningsmetod att tillämpas i projektet.

Den tekniska utformningen av spiralsökningen baseras på en Arkimedisk spiral. I

polära koordinater definieras Arkimedisk spiral enligt Ekvation 4.1.

$$r(t) = a + b \cdot \theta(t),\ \theta(t) \propto t \tag{4.1}$$

Här betecknar $r(t)$ avståndet från centrum vid tiden t, a startpunktens avstånd från centrum, b stigningskoefficienten som styr det konstanta avståndet och θ vinkeln vid tiden t. Viktigt att poängtera vid optimeringen av systemet är att stigningskoefficienten b måste vara mindre än svepbredd för drönarna, vilket är den area som drönaren kan detektera. Detta beror på att systemet ska minimera skuggytor. En annan viktig detalj är att vinkelhastigheten θ är proportionell mot tiden, detta är viktigt vid implementationen av drönare.

Implementation av en Arkimedisk spiral enligt det kartesiska koordinatsystemet definieras enligt Ekvation 4.2.

$$\boldsymbol{\gamma}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix},\ \ x(t) = r\cos(\theta),\ \ y(t) = r\sin(\theta),\ \ r, \theta \in \mathbb{R}. \tag{4.2}$$

4.1.2 Lissajouskurvor

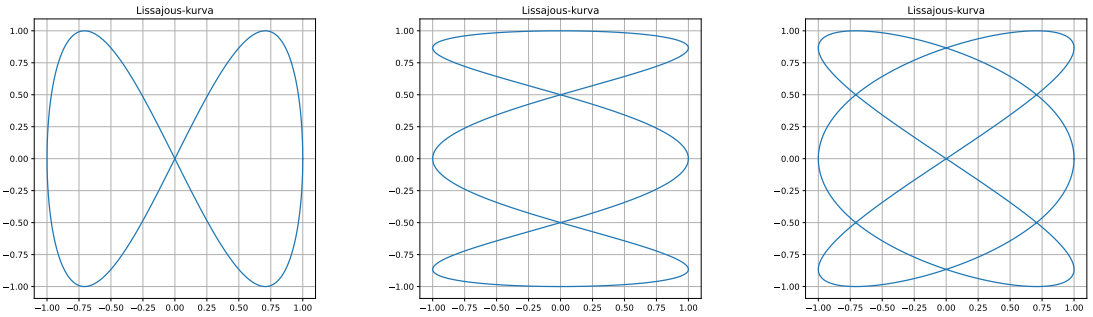
Ytterligare ett avsökningsmönster som anses relevant att studera är *Lissajous*. Avsökningsmönstret ger en snabb täckning samtidigt som det minimerar mekanisk belastning på drönarens hårdvara [12].

En Lissajouskurva är en typ av parametrisk kurva som kan användas för banplanering inom områdestäckning. En sådan kurva definieras av två sinusformade funktioner i respektive dimension i det tvådimensionella planet $\mathbb{R}^2$, vilket resulterar i en kontinuerlig och periodisk bana som täcker ett rektangulärt område över tid [12]. Kurvans form och huruvida den är sluten eller inte beror på förhållandet $\frac{A}{B}$. Om kvoten av $\frac{A}{B}$ är rationell är kurvan sluten, vilket i dessa sammanhang är relevant. Banan definieras enligt Ekvation 4.3.

$$\boldsymbol{\gamma}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix},\ \ x(t) = A\sin(at + \delta),\ \ y(t) = B\sin(bt),\ \ A, a, B, b, \delta \in \mathbb{R}. \tag{4.3}$$

Här representerar A och B amplituden i x- respektive y-led. a och b representerar vinkelfrekvenserna vilket påverkar områdets täthet och δ representerar fasförskjutning vilket avgör banans initiala form. I projektet användes amplituderna för att justera det område som Lissajouskurvan applicerades på. För att modifiera kurvba-

nans densitet går ändras vinkelfrekvenserna a och b, se Figur 4.1.



(a) $a = 1$, $b = 2$ (b) $a = 3$, $b = 1$ (c) $a = 3$, $b = 2$

Figur 4.1: Olika Lissajouskurvor för olika frekvensförhållanden.

Lissajouskurvor är glatta och därmed teoretiskt lämpliga för drönarflygning, detta eftersom behovet av stationär hovring vid kursändringar elimineras och därmed minimerar energiförbrukningen (se 2.2). I praktiken behöver hänsyn tas till mycket snäva kurvor som för en signifikant andel konfigurationer uppstår i områdets hörn. Det finns ett mervärde i att utforska vilken påverkan dessa hörn har på drönaren under flygning, men det ligger utanför ramarna för detta projekt.

En central egenskap hos Lissajouskurvor är att de är slutna och periodiska, vilket innebär att drönarna kan fortsätta att patrullera området utan behov av extra banplanering för att återvända till startpunkt. Genom att justera parametrarna i kurvans ekvation kan täckningsdensiteten kontrolleras, vilket möjliggör anpassning till olika sensorers räckvidd och krav på övervakning [12].

Vid användning av flera drönare kan dessa fördelas längs samma kurva med en viss fasförskjutning. Detta möjliggör simultan täckning av området samtidigt som kollisionskurser kan förutsägas och undvikas, eftersom kurvans skärningspunkter är deterministiskt definierade.

4.2 Framställning av banor

För att realisera en avsökning med *startTrajectory* krävdes en CSV-fil (eng. *Comma Separated Values*) som representerade avsökningsbanan. CSV är ett filformat som representerar data i tabellform, där varje rad består av numeriska värden som beskriver drönarens position i rummet. Därför konstruerades en banfilsgenerator med förmågan att utgå från en parametriserad kurva $\boldsymbol{\gamma}(t)$ som indata för att generera motsvarande instruktioner till drönarna.

4.2.1 Generering av banpunkter

För att realisera tidigare beskrivna avsökningsmönster (se avsnitt 4.1) implementerades tidsparametriska kurvor enligt Ekvation 4.4.

$$\boldsymbol{\gamma}(t) \mapsto \begin{bmatrix} x(t) & y(t) & z(t) & yaw(t) \end{bmatrix}^\top \tag{4.4}$$

De parametriska kurvorna användes för att skapa segment för en approximerad bana som definierar drönarens geometriska koordinater vid varje tidssteg t. Efter diskretisering av denna bana erhölls en punktmängd som utgör en högupplöst representation av den valda kurvan. Punktmängden användes därefter för banutjämning.

Sökytans anpassningsförmåga säkerställdes genom användningen av dynamiska styrparametrar. En central del av de dynamiska parametrarna är justering av bredden B och längden L i den tvådimensionella avsökningsytan. Därutöver användes linjeavstånd d_l för det spiralformade avsökningsmönstret för att kontrollera tätheten i mönstret. Motsvarande anpassning för Lissajous-geometrin sker med vinkelfrekvenserna a och b.

Utöver de dynamiska parametrarna inkluderades även den maaximalt tillåtna accelerationen a_{max} som en kritisk begränsning. Den maximala accelerationen användes vid anpassning av tidsaxel för att säkerställa att drönaren inte överskred drönarens fysiska begränsningar.

Tidsbaserad parametrisering av spiralsökning

För att möjliggöra en implementerbar beskrivning av spiralsökningen normaliserades tiden enligt Ekvation 4.5.

$$s = \frac{t}{t_{\text{tot}}} \tag{4.5}$$

Genom att normalisera tiden s kan spiralens expansion och rotation kontrolleras oberoende av den totala tiden t_{tot}. Den radiella skalningen r_{scale} och vinkeln $\theta(t)$ beräknas enligt Ekvation 4.6.

$$\begin{cases} r_{scale} = r_{\min} + (1 - r_{\min}) \cdot s \\ \theta(t) = 2\pi \cdot n_{\text{turns}} \cdot s \end{cases} \tag{4.6}$$

Antalet varv bestäms utifrån linjeavståndet d_l enligt Ekvation 4.7. Där $r_{\max}$ är radien till sökområdets inskrivna cirkel.

$$n_{\text{turns}} = \frac{r_{\max}(1 - r_{\min})}{d_l} \tag{4.7}$$

Antalet varv påverkar spiralens täthet i området, där ett lägre linjeavstånd resulterar i en tätare sökstruktur.

4.2.2 Banutjämning

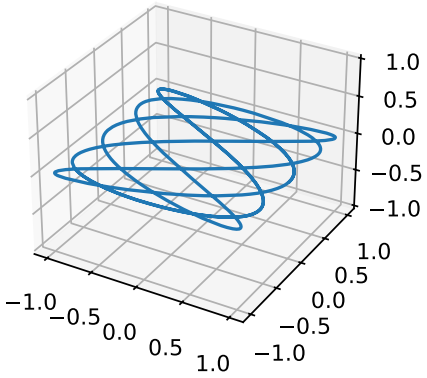
Den erhållna punktmängden utgjorde en diskret mängd, vilket inte är fysikaliskt realiserbart för en drönare att följa utan interpolation. Detta åtgärdades via polynomregression där minstakvadratmetoden användes för att anpassa ett sjundegradspolynom[1] som anpassades till punkterna. Sjundegradspolynom tillämpades för att möjliggöra kontinuerliga högre ordningens derivator, vilket bidrog till kontinuerliga banor.

Sjundegradspolynom valdes till följd av att den använda drönarplattformens mjukvara endast stödjer banor representerade som sjundegradspolynom sparade i en CSV-fil. Detta utgör av den orsaken en hård systembegränsning som gjorde valet polynom begränsad. För att erhålla en approximation av $\boldsymbol{\gamma}(t)$ användes segment av den ursprungliga kurvan, vars start- och slutpunkter bestämdes med ett tidssteg Δt. Polynomregressionen applicerades därefter på respektive segment.

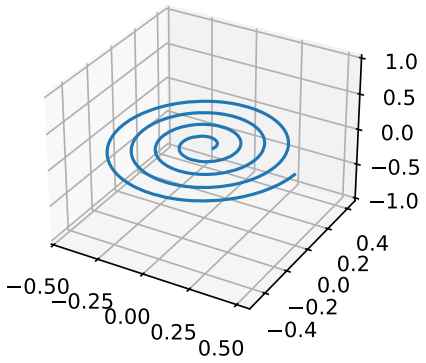
Koefficienterna ur en polynomregression lagrades därefter i en rad i CSV-filen, och på detta vis upprepades proceduren för varje segment tills det slutgiltiga mönstret återskapades.

Filerna var därefter redo att användas. Plottar av dessa kan ses i Figur 4.2 och 4.3.

[1] https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/trajectory_formats/, hämtad 12/5/2026.



Figur 4.2: Genererad Lissajouskurva.



Figur 4.3: Genererad spiralsökning.

4.2.3 Accelerationsbegränsare

Ett problem som upptäcktes med de genererade CSV-filerna var att den acceleration som krävdes för att drönaren skulle åstadkomma rutterna var obegränsad, vilket innebar att delar av trajektorierna riskerade att skada hårdvaran. För att åtgärda detta användes en skalfaktor s som bestämdes ur en önskad konstant $a_{\max}$ och den genererade trajektorien $\boldsymbol{r}$ enligt Sats 1 nedan.

Sats 1. *Låt $a_{max} > 0$ vara den högsta tillåtna accelerationen för en partikel längs en given parametriserad kurva $\boldsymbol{r}(t) \in C^2,\ t \in \mathbb{R}$.*

Det gäller då att skalfaktorn $0 < s < \sqrt{\dfrac{a_{max}}{\|\ddot{\boldsymbol{r}}(t)\|}}$ sådan att $\tilde{t} = st$ garanterar en acceleration lägre än a_{max}.

Bevis. Låt $\tilde{t} = st,\ s > 0$,
det följer att

$$\boldsymbol{r}(\tilde{t}) = \boldsymbol{r}(st).$$

Kedjeregeln två gånger ger

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{r}(\tilde{t}) = s^2\ddot{\boldsymbol{r}}(st).$$

Det är känt att $\ddot{\boldsymbol{r}}(t) = \dot{\boldsymbol{v}}(t) = \boldsymbol{a}(t)$, söker $\|\boldsymbol{a}(t)\| < a_{\max}$.

$$\|s^2\ddot{\boldsymbol{r}}(st)\| < a_{\max} \iff s^2 < \frac{a_{\max}}{\|\ddot{\boldsymbol{r}}(st)\|},\ \ \forall t$$

således erhålls

$$s < \sqrt{\frac{a_{\max}}{\|\ddot{\boldsymbol{r}}(t)\|}}.$$

□

För en globalt giltig skalfaktor beräknas alltså $s < \sqrt{\frac{a_{\max}}{\max_t \|\ddot{\boldsymbol{r}}(t)\|}}$.

4.2.4 Filstorleksbegränsningar vid banflygning

En kritisk teknisk begränsning med drönarmjukvaran som krävde proaktiv hantering var banfils-storleken. De drönare som användas i projektet hade en övre gräns för minnesallokering av banor på 4 kB, vilket ställer strikta krav på hur omfattande banfilerna får vara. Varje rad i den genererade CSV-filen motsvarar ett unikt polynomsegment vilket leder till att filstorleken är proportionerlig mot antalet segment i CSV-filen.

För att reducera filstorleken utfördes en optimering av banans diskretisering. Genom att öka tidslängden för varje enskilt segment minskades antalet segment som krävdes för att beskriva sökmönstret. Denna reducering innebär att färre rader behövdes läggas till i CSV-filen, vilket effektivt komprimerade filstorleken till en nivå som var kompatibel med drönarens minnesallokering.

Banans geometriska precision kunde bibehållas trots färre segment som en följd av användandet av sjundegradspolynom. Högre ordningens polynom medger en mer exakt anpassning över längre tidsintervall. Detta innebar att komplexa mönster kunde representeras inom minneskravet utan att kompromissa med banans mjukhet eller exakthet. Däremot är det fortfarande en avvägning mellan minnesutnyttjande och geometrisk trohet.

Sammantaget finns samtliga ovanstående delar implementerade i Algoritm 1. Implementationen återfinns i filen *csv_gen.py*, som finns tillgänglig i projektets gitlab-repo [13].

Algoritm 1 Trajectory planner

Require: Parametrized curve $\boldsymbol{\gamma}(t)$
Require: Maximum time $t_{\max}$
Require: Maximum acceleration $a_{\max}$
Require: Step size Δt
Require: Number of samples per segment n
Ensure: CSV file with polynomial coefficients

Compute acceleration limiting factor

$$s = \min\left(1,\ \sqrt{\frac{a_{\max}}{a_{\max,\boldsymbol{\gamma}}}}\right)$$

Initialize time

$t \leftarrow 0$

repeat

  $t_{\text{next}} \leftarrow \min(t + \Delta t,\ t_{\max})$

  Generate n equally spaced samples $\{t_i\}_{i=1}^{n} \subset [t,\ t_{\text{next}}]$

  for each sample $\{t_i\}_{i=0}^{n}$ **do**

    $\tilde{t} \leftarrow s t_i$

    $\begin{bmatrix} x_i & y_i & z_i & yaw_i \end{bmatrix} \leftarrow \boldsymbol{\gamma}(\tilde{t})$

  end for

  Define normalized time

$\tau_i \in [0, 1]$

  Fit seventh-degree polynomials to sampled points

Fit $\left(\begin{bmatrix} x(\tau) & y(\tau) & z(\tau) & yaw(\tau) \end{bmatrix}\right)$

  Add row of polynomial coefficients to CSV

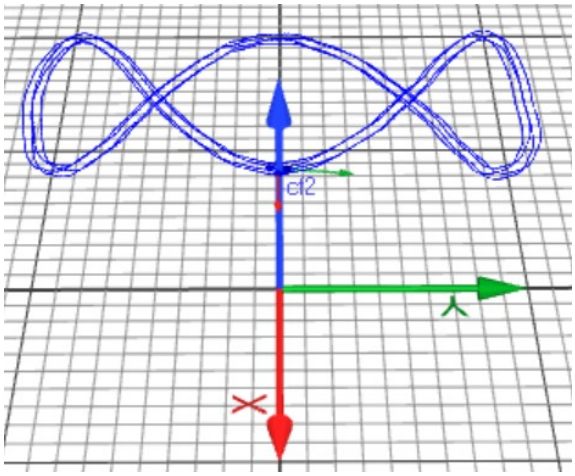
  $t \leftarrow t_{\text{next}}$

until $t \geq t_{\max}$

return CSV trajectory file modeling $\boldsymbol{\gamma}(t)$

4.3 Utvärdering av banflygning

Det är möjligt att implementera olika komplexa sökmönster, förutsatt att en parametriserad kurva γ som realiserar dem är känd. Dels i simulering och dels i verkligheten flyger drönarna enligt önskad bana på ett stabilt och mjukt sätt, se Figur 4.4. Däremot finns det begränsningar, bland annat förhindrar drönarnas minnesutrymme implementation av alltför komplexa banor, exempelvis banor med hög periodicitet. Det återstår för framtida arbeten att undersöka potentiella lösningar på dessa begränsningar.



Figur 4.4: Stillbild från inspelning av positionsdata vid test i verkligheten. Den blå banan visar positionshistorik för den drönare som flög enligt en Lissajouskurva från en genererad CSV-fil.

5

Detektion av inkräktaren

I detta kapitel behandlas delproblemet detektion av inkräktaren. Inledningsvis beskrivs hur ett synfält kan modelleras. Därefter lyfts projektets modellering och implementering av det simulerade synfältet med syftet att ge drönaren förmågan att detektera inkräktaren. Avslutningsvis utvärderas det implementerade synfältet.

5.1 Modellering av synfält

För att en drönare ska kunna detektera ett objekt krävs att den kan observera sin omgivning. Detta kan uppnås genom att utrusta drönaren med kameror eller andra sensorer, vilket ger upphov till ett synfält. Synfältet modelleras i [14], [15] och [16] med olika geometriska former, där en pyramid, kon respektive cirkel används.

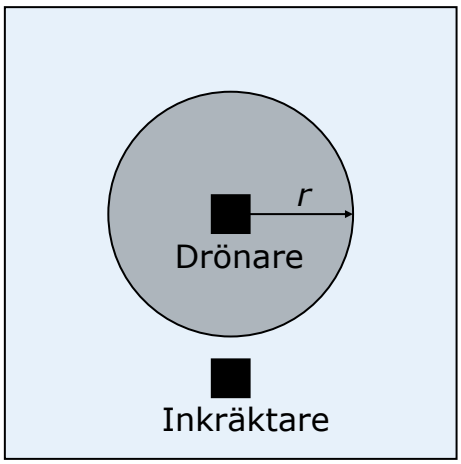
Markytan som vid en given tidpunkt anses täckt av drönaren definieras i både [14] och [15] som projektionen av den tredimensionella formen, en pyramid respektive en kon, på markplanet. Den täckta markytan beror därmed på drönarens flyghöjd. I [16] definieras den däremot som en cirkel med drönarens position som mittpunkt, vilket innebär att den täckta markytan är oberoende av flyghöjden.

5.2 Implementering av synfält

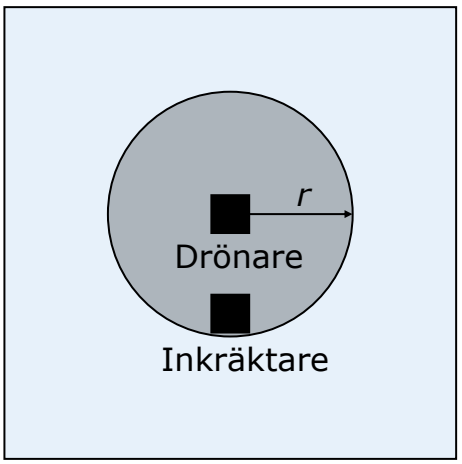
I projektet användes drönare som inte var utrustade med kameror eller andra sensorer som möjliggör observation av omgivningen. Drönarnas synfält behövde därför simuleras, vilket gjorde det nödvändigt att modellera synfälten.

För projektet var det mindre relevant att i hög grad efterlikna verkligheten. Synfälten modellerades därför på ett förenklat men fungerande sätt eftersom funktionalitet var väsentligt. Den geometriska form som ansågs lämpligast var en cylinder, eftersom modellen då, till skillnad från vid användning av en pyramid eller kon, blev oberoen-

de av flyghöjden vilket gjorde modellen enklare. Varje drönares synfält modellerades därför som en cylinder med radie r och den täckta markytan motsvarades av en cirkel med samma radie. Modellen antog således att det inte fanns några hinder som påverkade detektionsförmågan. Inkräktaren ansågs vara detekterad om den befann sig inom denna cirkel, vilket motsvarades av att avståndet mellan drönaren och inkräktaren i xy-planet var mindre än r, se Figur 5.1 och Figur 5.2.

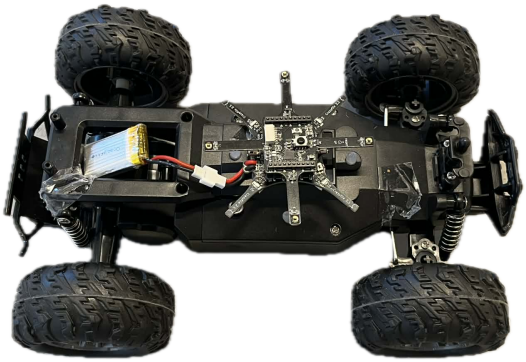


Figur 5.1: Inkräktaren befinner sig utanför drönarens synfält.



Figur 5.2: Inkräktaren befinner sig innanför drönarens synfält.

För att kunna bestämma avståndet mellan drönarna och inkräktaren krävdes dock att inkräktarens position var känd. Till detta nyttjades det befintliga MoCap-systemet. Ett Active marker deck monterades på en radiostyrd bil som agerade inkräktare vid fysiska tester, se Figur 5.3. Därigenom kunde inkräktarens position erhållas på samma sätt som drönarnas positioner.



Figur 5.3: Radiostyrd bil som agerade inkräktare vid fysiska tester, utrustad med ett Active marker deck.

Positionsinformationen från MoCap-systemet möjliggjorde implementeringen av drönarnas synfält. En ROS 2-nod skapades som lyssnade på ämnet där positionsinformationen publicerades. Denna information användes till att kontinuerligt avgöra om inkräktaren var detekterad eller inte, genom att beräkna om inkräktaren befann sig innanför någon av drönarnas modellerade synfält.

Beroende på om inkräktaren vid en given tidpunkt var detekterad eller inte lagrades relevant information i variabler som användes av styrskriptet vid beslutsfattande, se avsnitt 6.3. Om inkräktaren inte var detekterad sparades endast detta faktum, medan detektering medförde att även inkräktarens position samt namnet på drönaren som detekterade inkräktaren lagrades. Detta gjorde det möjligt att få drönaren som detekterat inkräktaren att byta från att avsöka området till att följa efter inkräktaren.

Ovanstående information, tillsammans med en tidsstämpel, publicerades på ett eget ämne. Detta för att möjliggöra kommunikation med det externa system som ansvarar för att omringa inkräktaren. Tidsstämplarna infördes för att underlätta felsökning, möjliggöra verifiering av korrekthet samt för att informationen vid behov ska kunna användas av det utomstående systemet.

För att möjliggöra testning med simulationsverktyget skapades även en snarlik ROS 2-nod för simulering av drönarnas synfält. Noden fungerade på samma sätt som

den tidigare nämnda, förutom att den lyssnade på ämnet där simulationsverktyget publicerade positionsinformationen.

Implementationen återfinns i filen *vision_node.py*, som finns tillgänglig i projektets gitlab-repo [13].

5.3 Utvärdering av synfältet

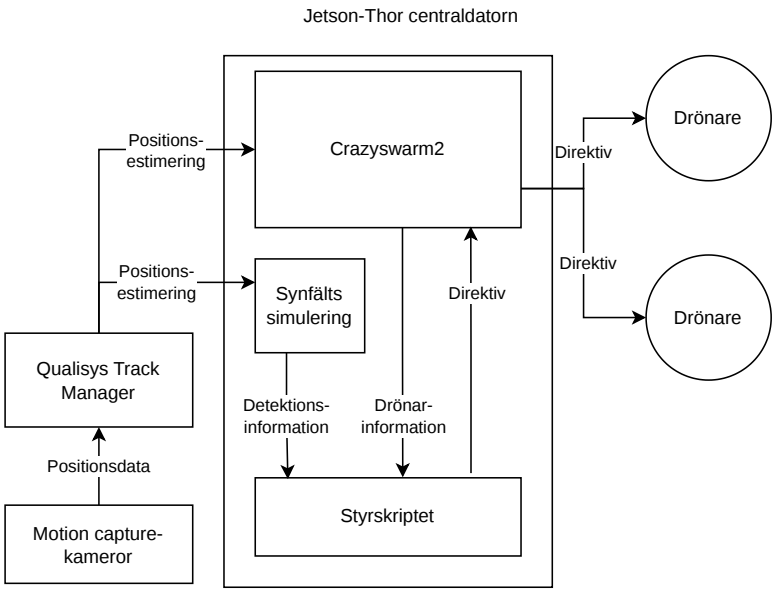
Den implementerade ROS 2-noden simulerade drönarnas synfält som cylindrar och publicerade relevant information på ett dedikerat ämne, vilket möjliggjorde att styrskriptet kunde fatta beslut gällande drönarstyrningen. Funktionaliteten verifierades genom observation av publicerade meddelanden.

En begränsning med implementationen är att om inkräktaren befinner sig vid utkanten av drönarens synfält kan detta leda till att noden växlar mellan att klassificera inkräktaren som detekterad respektive icke detekterad. Detta hanterades i styrskriptet, men en möjlig förbättring är att implementera denna hantering direkt i noden.

6

Arkitektur

I detta kapitel presenteras hur samtliga delkomponenter kommunicerar och hur de sammanfogades till ett komplett system, se Figur 6.1. Avslutningsvis presenteras styrskriptet med funktionalitet utifrån projektets syfte.



Figur 6.1: Översikt över systemets kommunikation

6.1 Positionsdata från MoCap-systemet

Positionsdatan finns kontinuerligt att hämta från en UDP-server som skapas vid en initial förfrågan till QTM. Därefter är positionsdata för varje i QTM definierad 6DoF-kropp, tillsammans med tillhörande rotationsmatris, tillgänglig och uppdaterad i realtid. Frekvensen av UDP-paketen med positionsdata är samma som kamerornas uppdateringsfrekvens.

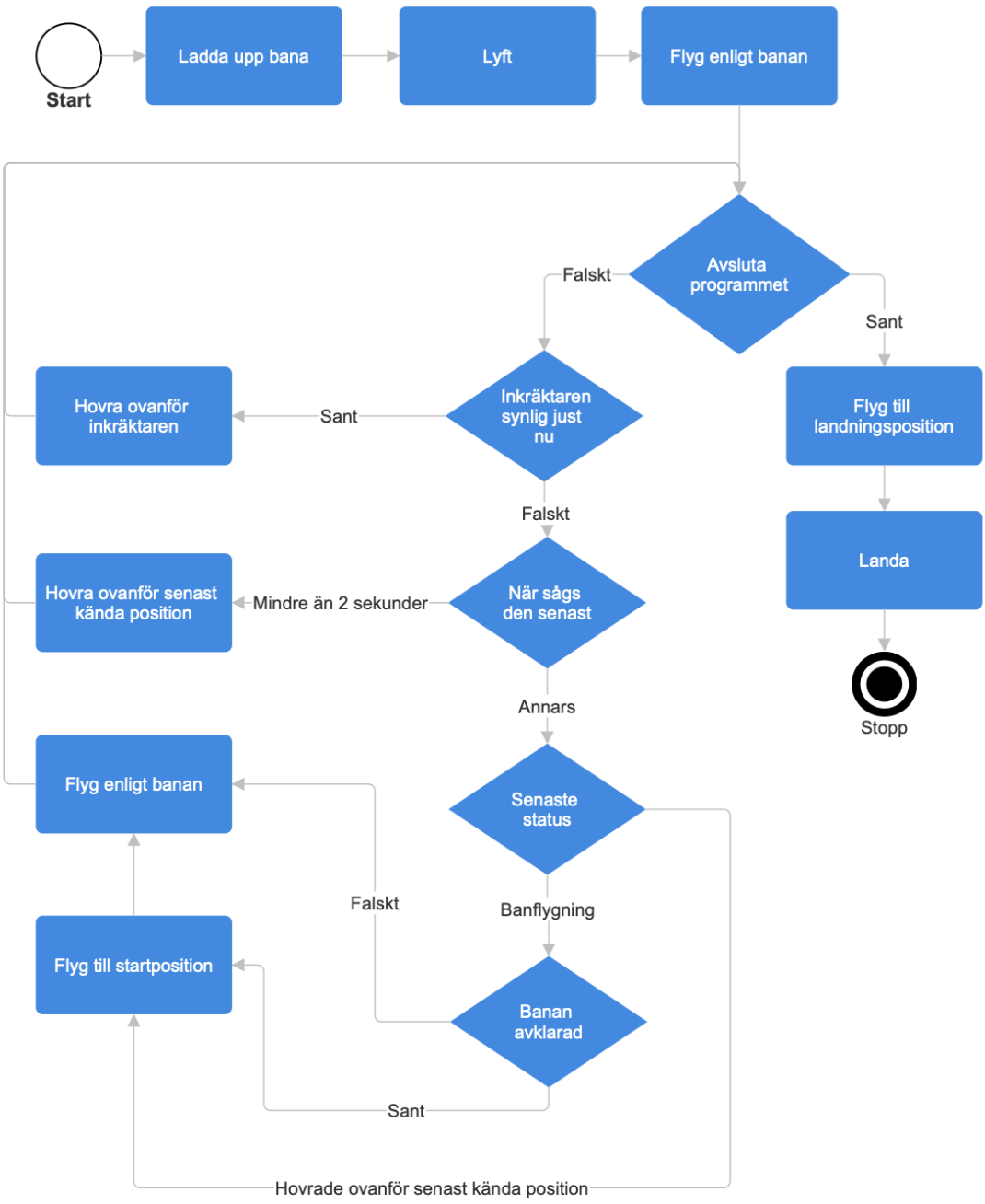
6.2 Vidarebefordran av positionsdata och externa drönardirektiv

Den initiala förfrågan om uppstarten av QTM UDP-servern samt vidarebefordran av den därefter realtidsuppdaterade positionsdatan till varje enskild drönare sker genom Crazyswarm2. Ansvaret att behandla och administrera externt inkommande drönardirekiv samt att kontinuerligt övervaka deras status görs även av Crazyswarm2 genom en 2.4 GHz radio.

6.3 Styrskriptet

Det framtagna styrskriptet, som exekverades på centraldatorn, integrerade de tidigare presenterade dellösningarna för att uppfylla projektets syfte. Skriptet växlade mellan avsökning och efterföljning baserat på detektionsinformationen från den nod som simulerade drönarnas synfält. Avsökningsbanorna som drönarna flög enligt skapades med banfilsgeneratorn. Styrningen av drönarna, vid både avsökning och efterföljning, möjliggjordes genom förflyttningsdirektiven, medan stabil flygning möjliggjordes av positionsdata från MoCap-systemet. I Figur 6.2 återfinns ett flödesschema som beskriver styrskriptets logik.

Vid implementeringen av styrskriptet utformades beteendet att en drönare som nyligen observerat inkräktaren, men därefter förlorat den ur sikte, inte omedelbart återgick till avsökning. För att möjliggöra återupptäckt, hovrade drönaren ovanför inkräktarens senaste kända position i maximalt två sekunder. Om drönaren inte observerade inkräktaren under denna tidsperiod skulle den återgå till sin avsökningsbana. Detta tillvägagångssätt valdes för att hantera den begränsning som nämnts i avsnitt 5.3. Implementationen återfinns i filen *trajectory_flyer.py*, som finns tillgänglig i projektets gitlab-repo [13].



Figur 6.2: Flödesschema över styrskriptets logik

7

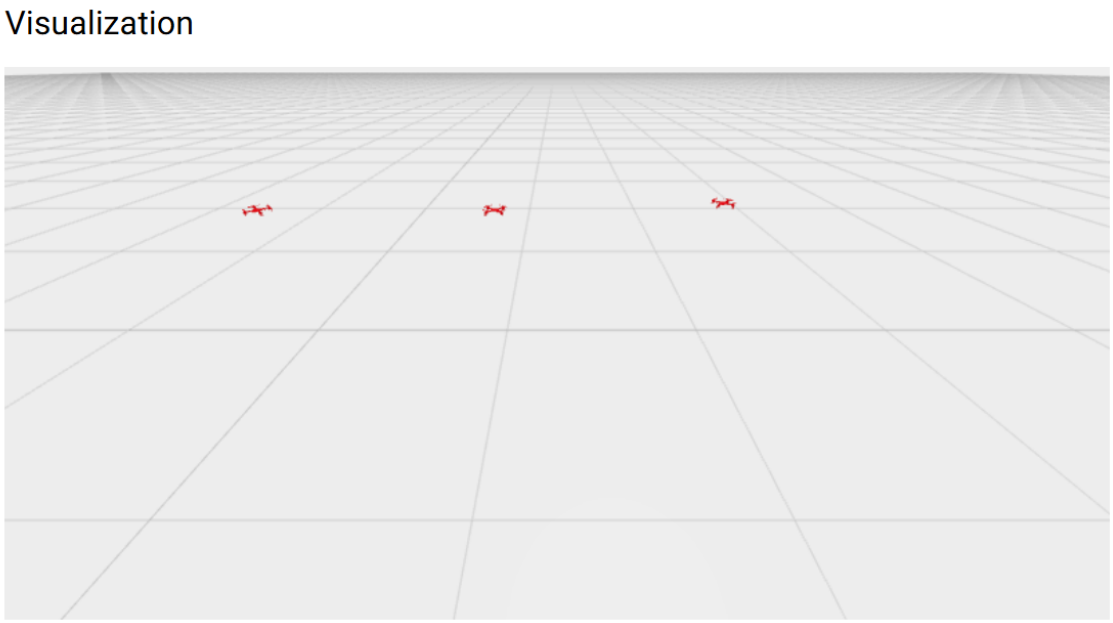
Resultat och diskussion

I detta kapitel presenteras resultaten från de simulerade och praktiska tester som genomfördes för att verifiera de lösningar som användes för projektets delproblem. Tillsammans utgör dessa lösningar ett system som uppfyller den funktionalitet som definierats i projektets syfte. Resultaten användes för att utvärdera systemets förmåga att genomföra områdesavsökning, detektera en inkräktare samt följa och vidarebefordra dess positionsdata efter detektion. De simulerade testerna användes främst för att verifiera banflygning, koordinering och tillståndsövergångar innan implementationen testades i fysisk miljö. Resultaten visar att flera delar av systemet fungerade enligt förväntan, samtidigt som ett antal begränsningar identifierades. Avslutningsvis diskuteras systemets styrkor samt möjliga förbättringar för de observerade begränsningarna.

7.1 Simulerade tester

Tester genomfördes initialt i simuleringsmiljö innan tester i verkligheten utfördes. Simuleringar användes för att utvärdera drönarnas rörelsemönster och stabilitet vid banflygning. I simulatorn flög drönarna enligt de genererade banorna med kontinuerliga och mjuka rörelser.

Simuleringsverktyget användes även för att verifiera koordinering av drönare. Genom att tilldela varje enskild drönare ett icke överlappande delområde, kunde flera drönare verka samtidigt utan risk för kollisioner under testerna. I Figur 7.1 visas en stillbild från testet.



Figur 7.1: Test av koordinerad banflygning för tre drönare med hjälp av simulationsverktyget

Vidare användes simuleringen för att testa övergången mellan områdesavsökning och efterföljning. Genom att låta en stillastående drönare agera inkräktare kunde tillståndsövergången verifieras genom att avbryta den pågående avsökningen om inkräktaren var inom drönarens synfält.

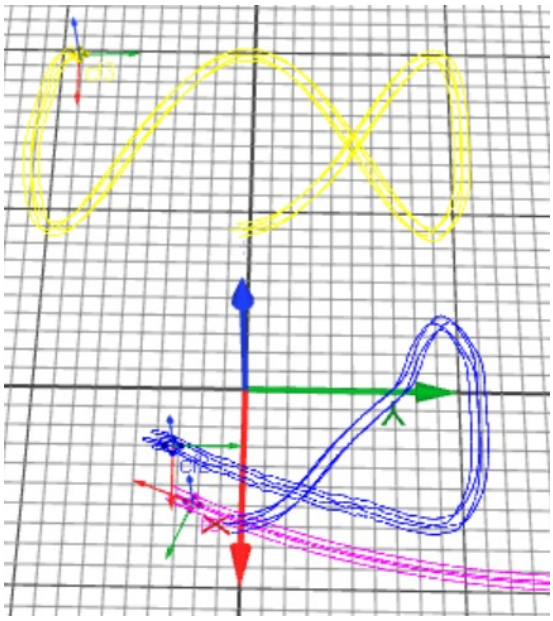
7.2 Fysiska tester

Efter initial verifiering i simuleringsmiljön genomfördes praktiska tester för att utvärdera systemets funktionalitet i fysisk miljö. Resultaten visade att drönarna kunde genomföra områdesavsökning samtidigt som stabil banflygning upprätthölls under större delen av testförloppet. Stundvis observerades instabil banflygning, med positionsfel och oscillationer runt önskad höjd.

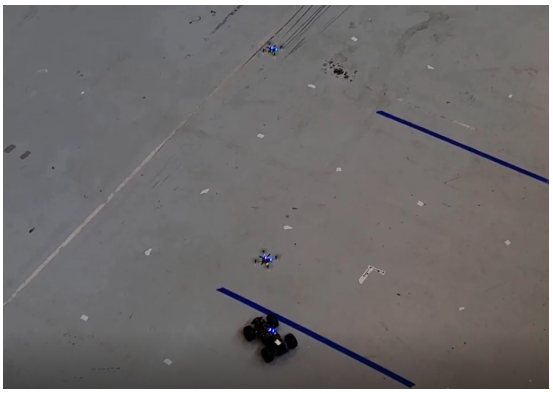
Detta framgår i Figur 7.2, där den gula drönaren uppvisade stabilare banflygning än den blå, trots att båda använde samma banfil och styrskript. Orsaken till skillnaden kunde inte fastställas, men observationer från samtliga tester som gjordes i den fysiska miljön indikerar att variationer i hårdvarans prestanda eller kvaliteten på MoCap-systemets positionsestimering kan ha påverkat flygstabiliteten.

Vid tester där en RC-bil agerade inkräktare kunde drönarna detektera och följa målet efter att det befunnit sig inom det simulerade detektionsområdet. Drönaren övergick då från områdesavsökning till efterföljning, vilket illustreras i Figur 7.2

och 7.3. Under efterföljningen vidarebefordrades positionsdata kontinuerligt till det externa systemet samtidigt som den andra drönaren fortsatte att genomföra områdesavsökning inom sitt tilldelade område.



Figur 7.2: Stillbild från inspelning av positionsdata vid test i verkligheten. De färgade banorna visar positionshistoriken för objekten. Gul representerar drönare ett, blå representerar drönare två, magenta representerar inkräktaren.



Figur 7.3: Övergång mellan områdesavsökning och målspårning i praktisk testmiljö. Den övre drönaren genomför avsökning enligt en Lissajousbana, den undre drönaren hovrar ovanför inkräktaren (RC-bilen) för att övervaka dess position.

Resultaten visar att systemet kunde genomföra områdesavsökning, detektion samt efterföljning av ett rörligt mål under kontrollerade förhållanden. Samtidigt är den nuvarande funktionaliteten begränsad till spårning av en enskild inkräktare, något som kan vidareutvecklas i framtida projekt.

Vidare saknar systemet förmågan att återgå till områdesavsökning efter detektion och efterföljning av inkräktaren, trots att sådan logik finns implementerat i styrskriptet. Problemet beror bedöms bero på svårigheter vid övergången mellan högnivåkommandon och strömningen av lågnivå setpoints, vilket ledde till att tillståndsbytet inte skedde korrekt och drönaren föll ner på marken. Detta utgör en begränsning i systemets robusthet och bör undersökas vidare i framtida projekt.

7.3 Framtida arbeten

Utöver de förbättringar av styrskriptet som presenteras i avsnitt 7.2 finns ytterligare utvecklingsområden som är intressanta att utforska i framtida arbeten.

På grund av drönarnas begränsade interna lagringsutrymme skulle komprimering av banfiler möjliggöra mer komplexa avsökningsmönster än de som presenterats inom ramen för detta arbete.

Möjligheten att anpassa den genererade banfilens område utifrån dynamiskt förändrade målområden skulle kunna öka projektets modularitet och effektivitet. Det skulle även möjliggöra dynamisk anpassning av antalet drönare som är aktiva inom området.

8

Slutsats

Studien visar att det är tekniskt genomförbart att utveckla ett autonomt drönarsvärmssystem för sök- och följningsuppgifter baserat på ett MoCap-system. Studien identifierade flera testmiljörelaterade begränsningar som påverkade drönarsystemets precision och funktionalitet, däribland kamerareflektioner och kameraplacering. Vidare konstaterades att drönarnas begränsade lagrings- och minneskapacitet ställde krav på optimering av bangenerering.

Det utvecklade systemet, baserat på de lösningar som togs fram för arbetets delproblem, verifierades genom fysiska tester där drönarna framgångsrikt utförde uppgiften under kontrollerade former. Samtidigt identifierades problem med tilltänkt funktionalitet, särskilt vid förlust av visuell kontakt med inkräktaren, vilket ledde till att drönaren kraschade. Vad detta berodde på är ej klarlagt och bör undersökas i framtida projekt.

Sammanfattningsvis bedöms projektet syfte ha uppfyllts genom utvecklingen och valideringen av ett fungerade drönarsvärmssystem. Projektets resultat visar på behovet av fortsatt utveckling inom området för autonoma drönarsvärmar, speciellt inom ökad feltolerans samt optimering av avsökningsstrategier för dynamiska miljöer.

Litteratur

[1] P. Majumdar, S. Mitra, · . Diptendu Bhattacharya och B. Bhushan, ”Enhancing sustainable 5G powered agriculture 4.0 : Summary of low power connectivity, internet of UAV things, AI solutions and research trends”, *Multimedia Tools and Applications*, årg. 84, s. 17 389–17 433, 2025. DOI: 10.1007/s11042-024-19728-1. URL: https://doi.org/10.1007/s11042-024-19728-1.

[2] V. Spinko. ”Drone swarms: How they actually work and what industries should care”. Besökt 12 maj 2026, WTWH Media LLC. URL: https://www.therobotreport.com/drone-swarms-how-they-actually-work-and-what-industries-should-care/.

[3] A. A. Kareem, A. J. Abid, D. A. Hammood, R. Alroobaea och A. Yousef, ”Enhancing search and rescue missions for victim detection using a coordinated bio-inspired dual-altitude UAV swarm framework”, *Engineering Science and Technology, an International Journal*, årg. 77, s. 102 353, maj 2026, ISSN: 2215-0986. DOI: 10.1016/J.JESTCH.2026.102353.

[4] M. Idrissi, H. Mahmoud, A. Wahab och J. M. Al Ja'Am, ”Decentralised Swarm Drones: Shaping the Future of Autonomous Secure Unmanned Aerial System”, i *2025 International Conference on Computer and Applications (ICCA)*, 2025, s. 1–6. DOI: 10.1109/ICCA66035.2025.11430889.

[5] M. Cella, F. d'Apolito, P. Fanta-Jende och C. Sulzbachner, *FUELING GLOCAL: OPTIMIZATION-BASED PATH PLANNING FOR INDOOR UAVS IN AN AUTONOMOUS EXPLORATION FRAMEWORK*, English, maj 2023. DOI: 10.5194/isprs-archives-XLVIII-1-W1-2023-85-2023. URL: https://research.ebsco.com/linkprocessor/plink?id=2f203a79-86fd-3031-93a7-3edd99808ccb.

[6] S. Macenski, T. Foote, B. Gerkey, C. Lalancette och W. Woodall, ”Robot Operating System 2: Design, architecture, and uses in the wild”, *Science Robotics*, årg. 7, nr 66, maj 2022, ISSN: 24709476. DOI: 10.1126/SCIROBOTICS.ABM6074.

[7] J. A. Preiss, W. Honig, G. S. Sukhatme och N. Ayanian, ”Crazyswarm: A large nano-quadcopter swarm”, *Proceedings - IEEE International Conference on Robotics and Automation*, s. 3299–3304, juli 2017, ISSN: 10504729. DOI: 10.1109/ICRA.2017.7989376.

[8] *Choice of markers*. URL: https://docs.qualisys.com/qtm/content/running_the_system/ir_markers.htm.

[9] *Qualisys active markers*. URL: https://docs.qualisys.com/qtm/content/running_the_system/qualisys_active_markers.htm.

[10] *The Commander Framework | Bitcraze*. URL: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/commanders_setpoints/.

[11] IMO och ICAO, *IAMSAR manual : international aeronautical and maritime search and rescue manual. Volume III Mobile facilities.* eng, 9th ed., 2016 ed. IMO, jan. 2016, vol. III, ISBN: 9789280116397. URL: https://research.ebsco.com/linkprocessor/plink?id=bdfa97e9-88c4-37c2-a850-7c13ed45b4ba.

[12] A. V. Borkar, A. Sinha, L. Vachhani och H. Arya, ”Application of Lissajous curves in trajectory planning of multiple agents”, *Autonomous Robots*, årg. 44, nr 2, s. 233–250, jan. 2020, ISSN: 15737527. DOI: 10.1007/S10514-019-09888-7. URL: https://openurl.ebsco.com/contentitem/doi:10.1007/s10514-019-09888-7?sid=ebsco:plink:crawler&id=ebsco:doi:10.1007/s10514-019-09888-7&crl=c.

[13] Ninus Sellin Bredberg, Alma Edling, Oscar Larsson, Jesper Skoglund och Hassan Ahmad, *Kandidatarbete EENX16 · GitLab*. URL: https://git.chalmers.se/ninuss/kandidatarbete-eenx16.

[14] Y. Wu och K. H. Low, ”Route Coordination of UAV Fleet to Track a Ground Moving Target in Search and Lock (SAL) Task Over Urban Airspace”, *IEEE Internet of Things Journal*, årg. 9, nr 20, s. 20 604–20 619, okt. 2022, ISSN: 23274662. DOI: 10.1109/JIOT.2022.3178089. URL: https://ieeexplore.ieee.org/document/9782678.

[15] S. Papatheodorou, A. Tzes och Y. Stergiopoulos, ”Collaborative visual area coverage”, *Robotics and Autonomous Systems*, årg. 92, s. 126–138, juni 2017, ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2017.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0921889016306960?via%3Dihub.

[16] K. Kim och J. Kim, "Coordinated Informative Path Planning for Multi-Robot Search in Open Fields", *Journal of Intelligent & Robotic Systems*, årg. 111, nr 2, maj 2025, ISSN: 1573-0409. DOI: 10.1007/S10846-025-02270-Z. URL: https://link.springer.com/article/10.1007/s10846-025-02270-z.

INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS