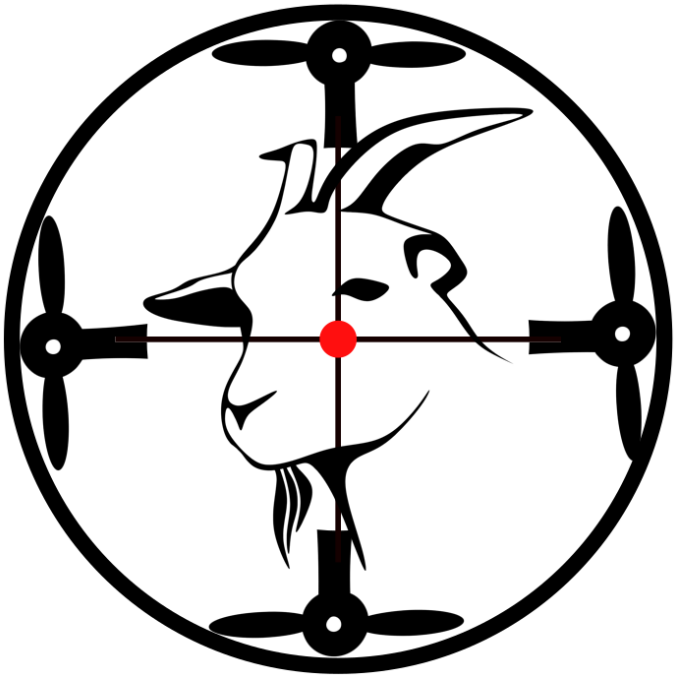
Designspecifikation
Följning av djur Kolmården djurpark

Version 1.0

Projektgrupp: Tar-Get
2017-10-11



Status

Granskad	JE, HL	2017-10-10
Godkänd	Beställare	2017-10-10

PROJEKTIDENTITET

2017/HT, Följning av djur Kolmården djurpark
Linköpings Universitet, ISY

Gruppdeltagare

Namn	Ansvar	Telefon	E-post
Daniel Arnström	Mjukvaruansvarig	076-8312409	danar892@student.liu.se
Adam Bergenhem	Integrationsansvarig	073-2306520	adabe471@student.liu.se
Joakim Ekström	Hårdvaruansvarig	070-3312603	joaek309@student.liu.se
Tim Fornell	Designansvarig	072-3949005	timfo734@student.liu.se
Jacob Holmberg	Dokumentansvarig	070-3915033	jacho926@student.liu.se
Henrik Lillberg	Projektledare	072-7331600	henli757@student.liu.se
Gustav Magnusson	Leveransansvarig	070-2876287	gusma061@student.liu.se
Peter Mrad	Testansvarig	076-2008523	petmr615@student.liu.se
Johan Svensson	Informationsansvarig	070-9187399	johsv660@student.liu.se

E-postlista för hela gruppen: henli757@student.liu.se
Hemsida: www.isy.liu.se/edu/projekt/tsrt10/2017/kolmarden

Beställare: Christian A. Naesseth, Linköpings Universitet
Telefon: +46 13 281087, Mail: christian.a.naesseth@liu.se
Kund: Gustaf Hendeby, Linköpings Universitet
Telefon: +46 13285815, Mail: gustaf.hendeby@liu.se
Kursansvarig: Daniel Axehill, Linköping Universitet
Telefon: +46 13 284042, Mail: daniel@isy.liu.se
Handledare: Fredrik Ljungberg, Linköpings Universitet
Mail: fredrik.ljungberg@liu.se

Innehåll

Dokumenthistorik **1**

1 Inledning **2**
1.1 Parter . . . 2
1.2 Projektets bakgrund . . . 2
1.3 Syfte och mål . . . 2
1.3.1 Kortsiktiga mål . . . 2
1.3.2 Långsiktiga mål . . . 2
1.4 Användning . . . 3
1.5 Definitioner . . . 3

2 Översikt **4**
2.1 Hårdvara . . . 5
2.2 Mjukvara . . . 5
2.3 Koordinatsystem . . . 6

3 Kommunikation **8**
3.1 Topics . . . 8
3.2 Services . . . 9

4 GUI **11**
4.1 Gränssnitt . . . 12

5 Bildbehandling **13**
5.1 Övergripande design . . . 13
5.2 Terminologi . . . 14
5.3 Förbehandling . . . 14
5.4 Detektera mål . . . 15
5.5 Markering av mål . . . 15
5.6 Omvandling av koordinater . . . 15
5.6.1 Konvertering av bildkoordinater . . . 15
5.6.2 Konvertering till lokalt koordinatsystem . . . 16
5.7 Gränssnitt . . . 18

6 Positionering **19**
6.1 Gränssnitt . . . 19

7 Målföljning **21**
7.1 Övergripande design . . . 21
7.2 Rörelsemodell . . . 22
7.3 Tidsuppdatering . . . 23
7.4 Associering . . . 23
7.4.1 Avståndsberäkning . . . 23
7.4.2 Nearest Neighbor . . . 23
7.5 Mätuppdatering . . . 24

7.6 Gränssnitt 24

8 Uppdragsplanering **25**
8.1 Övergripande design 25
8.2 Uppdrag 1 27
8.3 Uppdrag 2 27
8.4 Uppdrag 3 28
8.5 Gränssnitt 28

9 Simulering **29**

Dokumenthistorik

Version	**Datum**	**Utförda förändringar**	**Utförda av**	**Granskad**
0.1	2017-09-15	Första utkast.	Projektgruppen	DA, JH
0.2	2017-09-18	Komplettering efter kommentar från handledare.	JH, JS, JE, PM, TF, HL, GM, DA	JE, DA
0.3	2017-09-29	Första utkast inför BP3.	Projektgruppen	AB, GM, PM
0.4	2017-10-04	Kompletterad efter kommentarer från handledare.	Projektgruppen	JS
0.5	2017-10-06	Kompletterad efter ytterligare kommentarer från handledare.	Projektgruppen	TF, JH, JS
1.0	2017-10-10	Kompletterad efter kommentarer från beställare vid BP3.	Projektgruppen	JE, HL

1 Inledning

Det här dokumentet är en designspecifikation för projektet ”Följning av djur i Kolmården djurpark”. Dokumentet är ett examinationsmoment i kursen TSRT10 Reglerteknisk projektkurs som ges av ISY vid Linköpings universitet under höstterminen 2017. Designspecifikationen beskriver systemet och de övergripande lösningarna som är tänkt att implementeras för att produkten skall uppfylla kraven listade i kravspecifikationen.

1.1 Parter

Projektet innehåller följande parter

- Kund: Gustaf Hendeby, ISY
- Beställare: Christian Andersson Naesseth, ISY
- Handledare: Fredrik Ljungberg, ISY
- Examinator: Daniel Axehill, ISY
- Projektgruppens medlemmar

1.2 Projektets bakgrund

Bakgrund till projektet är att hjälpa parkvakter i Ngulia, Kenya, att skydda noshörningar i dess noshörningsreservat. Med hjälp av drönare ska parkvakterna kunna få bättre förmåga att upptäcka inkräktare och ge en djupare förståelse för hur många djur reservatet innehåller samt djurens rörlsemönster. För att på ett mer lättillgängligt sätt kunna testa tekniken finns ett samarbete med Kolmården.

1.3 Syfte och mål

Syftet med projektet är att projektmedlemmarna ska tillämpa tidigare inlärd kunskap inom området och använda sig av den praktiskt. Dessutom ska arbetet ske enligt projektmodellen LIPS för att få ett tillvägagångsätt som är snarlikt arbetslivet. Projektets mål delas in i kortsiktiga samt långsiktiga.

1.3.1 Kortsiktiga mål

Målet med projektet är att bygga ett system som kan detektera och följa djur. Systemet består av en drönare utrustad med kamera, GPS-mottagare och tillhörande mjukvara.

1.3.2 Långsiktiga mål

Ett långsiktigt mål är att utveckla en plattform som kan övervaka ett större område autonomt från luften. I första hand kan systemet tänkas användas för demonstration av tekniken på Kolmården, och på längre sikt användas för att hjälpa parkvakter i Kenya.

1.4 Användning

Med de tidsramar som finns är slutprodukten tänkt att kunna söka av ett område och identifiera mål i kombination med att följa på flera mål samtidigt. Produkten kan komma att användas som en grund för vidareutveckling och till slut användas i Kenya för att autonomt övervaka noshörningsreservat.

1.5 Definitioner

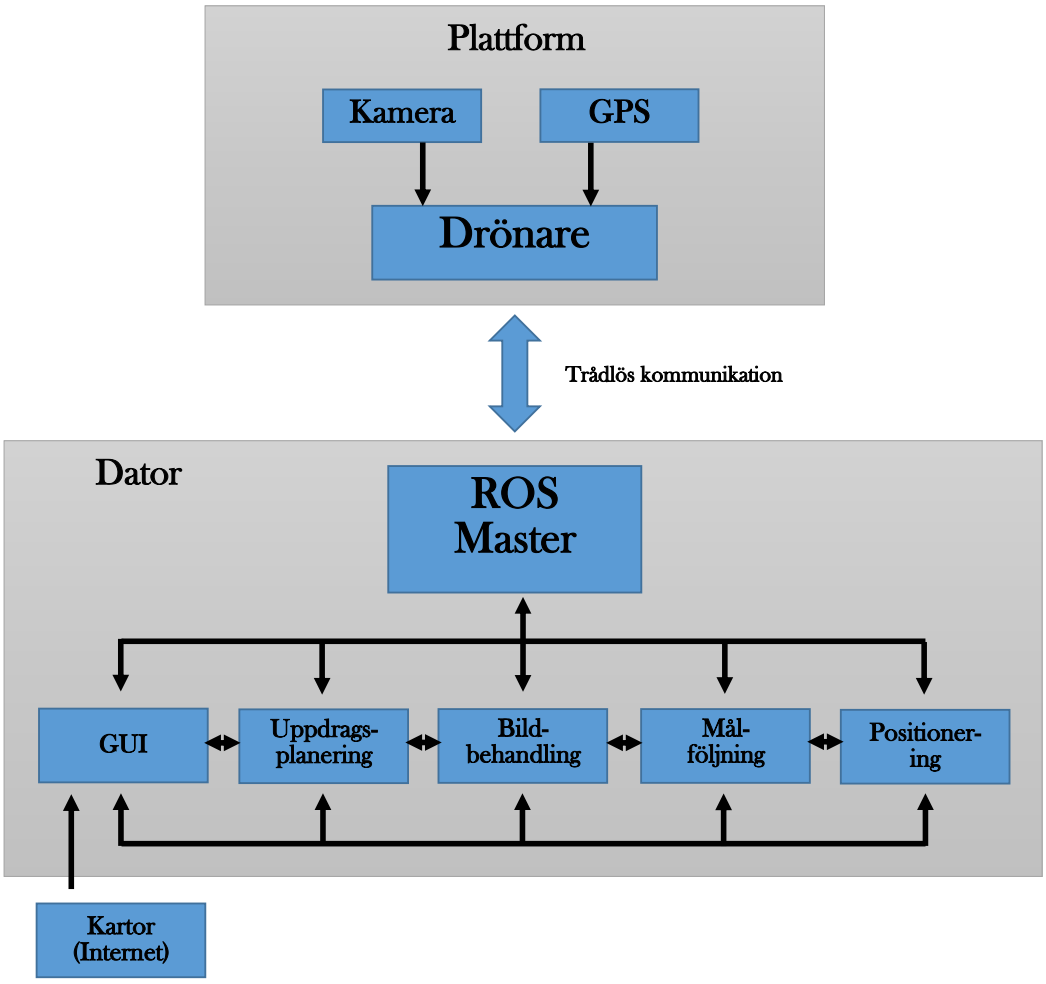
En beskrivning av de begrepp som används i dokumentationen visas nedan.

- ROS: Robot Operating System.
- Plattform: Med detta menas quadcoptern med tillhörande hårdvara.
- Mål: Objekt som försöker lokaliseras av systemet.
- Produkt: Quadquoptern med tillhörande mjukvara som körs på en dator.
- Gazebo: Simuleringsverktyg.

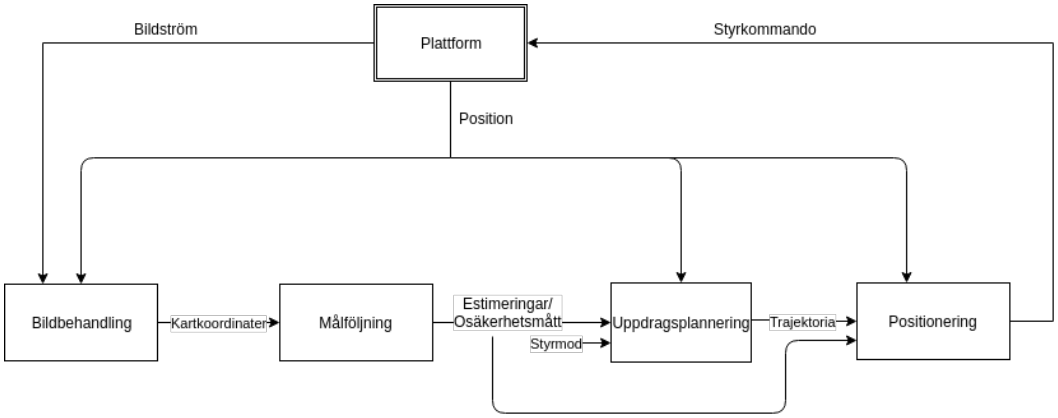
2 Översikt

Produkten av projektet ska vara en drönare som, både manuellt och autonomt, kan söka av ett valt område och identifiera mål. Utöver detta skall även målföljning finnas tillgängligt. Till en början kommer stillastående enkla mål detekteras, men ambitionen är att rörliga och mer svårdetekterade mål (getter) skall kunna identifieras.

Systemet i helhet ses i figur 1, där blocken motsvarar moduler eller komponenter. Pilarna visar hur kommunikation mellan dessa sker. Modulerna inkluderar GUI, uppdragsplanering, bildbehandling, målföljning samt positionering. Modulerna kommunicerar med varandra via ramverket ROS. I figur 2 visas ett översiktligt blockschema för systemet.



Figur 1: Skiss över systemets uppbyggnad och ingående moduler.



Figur 2: Översiktligt blockschema för systemet

2.1 Hårdvara

Hårdvaran som används i projektet är en DJI Matrice 100 drone med visuell och/eller infraröd kamera. Leveransdatum av DJI Matrice 100 är inte specificerat och om ordinarie hårdvara ej fås inom tidsramarna kommer en AR Drone 2.0 användas för demonstration. Projektgruppen kommer att använda sina egna datorer.

2.2 Mjukvara

ROS används som ramverk för att köra mjukvaran från systemets delsystem. Varje delsystem bildar en ROS-nod som sedan kan köras oberoende av systemets andra noder. Systemets noder är:

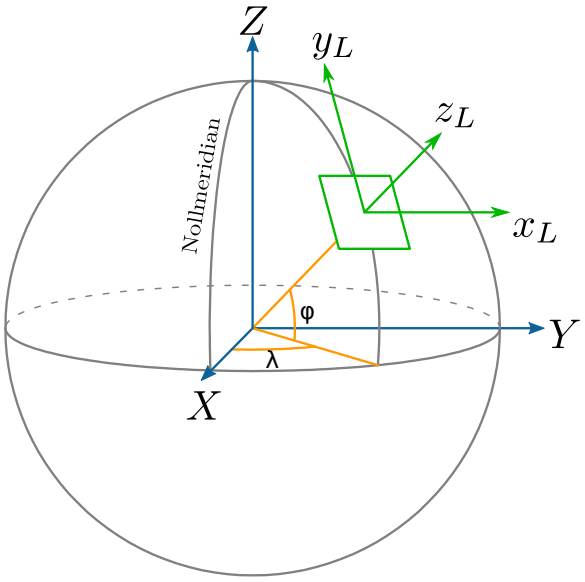
- **roscore**: Har som uppgift att bland annat övervaka övriga noder.
- **tracking**: Har som uppgift att skatta positioner för detekterade objekt.
- **image_processing**: Har som uppgift att detektera objekt i den videoström som fås från kameran på drönaren.
- **planning**: Har som uppgift att planera hur plattformen ska röra sig för att utföra ett givet uppdrag optimalt.
- **positioning**: Ska se till att drönaren följer trajektorian som tas fram av uppdragsplaneringen.
- **gui**: Gör det möjligt för användaren att kommunicera med samt se information om plattformen.
- **dji_sdk**: Ska bland annat skicka videoströmmen från kameran, data från GPS:en och statusinformation om drönaren till datorn samt ta emot styrkommandon från datorn.

Koden till alla delsystem ska skrivas i C++ om inget annat angivits.

2.3 Koordinatsystem

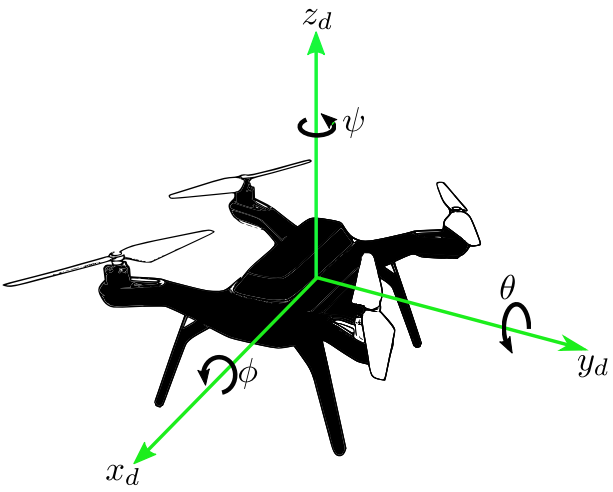
Tre olika koordinatsystem kommer användas av modulerna. Ett koordinatsystem (X, Y, Z) som har origo i jordens centrum. I detta koordinatsystem kommer koordinater anges i longitud λ och latitud φ och radien motsvarar jordens radie.

Ett kartesiskt koordinatsystem (x_L, y_L, z_L) med origo där plattformen startar sin flygning. Detta koordinatsystem är ett ENU-koordinatsystem vilket innebär att x_L, y_L och z_L kommer vara riktad åt öst, norr och upp och kommer att refereras till som det ”lokala koordinatsystemet”. Hur koordinatsystemet relateras till jordens koordinatsystem kan ses i figur 3.



Figur 3: Förhållandet mellan ett fixt koordinatsystem i jorden och ett lokalt ENU-system.

Slutligen kommer ett kartesiskt koordinatsystem (x_d, y_d, z_d) vara fäst i drönaren. Plattformens rotation relativt (x_L, y_L, z_L) definieras av eulervinklarna (ϕ, θ, ψ) och motsvarande rotationsmatris $R = R_\phi^x R_\theta^y R_\psi^z$ där ψ är girvinkeln, θ är tiltvinkel och ϕ är rollvinkel. Dessa vinklar beskriver hur mycket drönarens koordinatsystem (x_d, y_d, z_d) har roterat relativt det lokala koordinatsystemet (x_L, y_L, z_L). Figur 4 illustrerar hur drönarens koordinatsystem samt dessa vinklar är definierade.



Figur 4: Drönarens lokala koordinatsystem.

3 Kommunikation

Kommunikationen sker med hjälp av ramverket ROS. ROS-noderna kan kommunicera med varandra genom att publicera och prenumerera på meddelanden från rostopics. Flera ROS-noder kan publicera och prenumerera på samma topic. Kommunikation via topics är till för informationsflöden som är kontinuerliga. Om man istället vill göra enstaka efterfrågningar mellan moduler kan istället rosservices användas. Ros-noder kommunicerar via services genom att en nod begär en service och en annan nod tillhandahåller denna service.

3.1 Topics

Tabell 1 visar systemets topics, vilka noder som prenumererar/publicerar på respektive topic samt vilken meddelandetyp som publiceras på respektive topic. För en utförlig beskrivning av standardmeddelanden från *std_msgs*, *geometry_msgs*, och *sensor_msgs* se [1][2]. Notera även att bara de mest relevanta topics från dji_sdk listas. För att se samtliga topics som dji_sdk prenumererar/publicerar till se [3].

Tabell 1: ROS-topics för systemet.

Topic	**Publishers**	**Subscribers**	**Meddelandetyp**
/goal_estimations	tracking	planning positioning gui	GoalEstimateArray
/detection/map_coordinates	image_processing	tracking	geometry_msgs/Point
/detection/image_coordinates	image_processing	gui	geometry_msgs/Point
/planned_trajectory	planning	positioning	PointArray
/search_area	gui	planning	PointArray
/mission_state	gui	planning	std_msgs/UInt8
/dji_sdk/local_position	dji_sdk	image_processing planning positioning gui	geometry_msgs/PointStamped
/dji_sdk/imu	dji_sdk	image_processing	sensor_msgs/Imu
/dji_sdk/image_raw	dji_sdk	image_processing gui	sensor_msgs/Image
/dji_sdk/gps_health	dji_sdk	image_processing planning positioning gui	std_msgs/UInt8
/dji_sdk/battery_state	dji_sdk	gui	sensor_msgs/BatteryState

Nedan beskrivs vad varje topic innehåller för information.

- **/goal_estimations**
  Positionsestimeringar för mål med en kovarians för varje estimering. Positionerna ges i ett lokalt kartesiskt koordinatsystem.

- **/detection/map_coordinates**
  Kartkoordinater för ett detekterat mål. Positionen ges i ett lokalt kartesiskt koordinatsystem.
- **/detection/image_coordinates**
  Bildkoordintar för ett detekterat mål.
- **/planned_trajectory**
  Innehåller en array med punkter, som motsvarar positioner i ett lokalt kartesiskt koordinatsystem, som plattformen önskas följa.
- **/search_area**
  En array med punkter som definierar ett sökområde som plattformen ska söka av.
- **/mission_state**
  Ett heltal som representerar vilket uppdrag som plattformen ska utföra. Tabell 2 visar vilket värde på heltalet som motsvarar vilken uppdragstyp.

Tabell 2: Värde på mission_state som motsvarar uppdragstyp

Värde	Uppdragstyp
0	Manuell styrning
1	Uppdrag 1
2	Uppdrag 2
3	Uppdrag 3

- **/dji_sdk/local_position**
  Plattformens nuvarande position som anges i ett lokalt kartesiskt koordinatsystem.
- **/dji_sdk/imu**
  Mätvärden från plattormens IMU. Innehåller plattformens orientation, vinkelhastighet och hastighet.
- **/dji_sdk/image_raw**
  Bildström från plattformens kamera.
- **/dji_sdk/gps_health**
  Anger kvalitén på signalstyrkan för plattformens GPS. Kvalitén representeras av ett heltal mellan 0 och 5; där 5 innebär god signalstyrka.
- **/dji_sdk/battery_state**
  Anger status för batteriet. Innehåller bland annat ett flyttal mellan 0 och 1 som representerar batteriprocenten.

3.2 Services

Tabell 3 visar systemets services, vilka noder som tillhandahåller/begär dessa och vilket typ som servicen är. Notera att bara de mest relevanta servicen från dji_sdk listas. För att se samtliga services som dji_sdk tillhandahåller se [3].

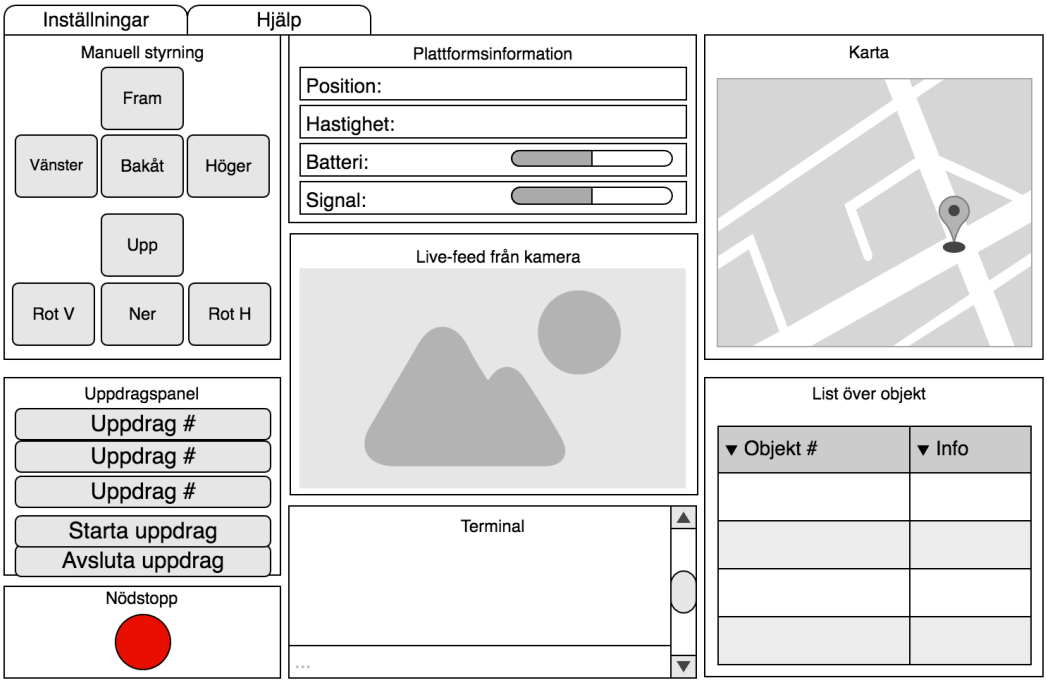
Tabell 3: ROS-services för systemet.

Service	**Provider**	**Requester**	**Service type**
/dji_sdk/mission_waypoint_upload	dji_sdk	positioning	dji_sdk/MissionWpUpload
/dji_sdk/mission_waypoint_action	dji_sdk	positioning	dji_sdk/MissionWpAction
/dji_sdk/sdk_control_authority	dji_sdk	positioning	dji_sdk/SDKControlAuthority

- **/dji_sdk/mission_waypoint_upload**
  Anger ett waypoint-uppdrag som plattformen ska utföra. Uppdraget består av waypoints som plattformen ska besöka samt andra parametrar som exempelvis flyghastighet. En waypoint anges i latitud, longitud och altitud.
- **/dji_sdk/mission_waypoint_action**
  Servicen startar, stoppar, pausar eller återupptar ett waypoint-uppdrag.
- **//dji_sdk/sdk_control_authority**
  Används för att begära att få kontroll över plattformens styrning. Returnerar sant eller falskt beroende på om begäran accepteras eller nekas.

4 GUI

Denna modul kommer att sköta all interaktion mellan användaren och systemet, en skiss över hur den ska se ut kan ses i figur 5. GUI:t ska skapas med hjälp av utvecklingsmiljön Qt [4] och kartorna som behövs kommer att hämtas med Marble [5]. ROS-paketet rqt [6] kommer användas för att implementera GUI:t i ROS.



Figur 5: Skiss över uppbyggnaden av GUI:t.

GUI:t kommer att innehålla följande:

- En karta där positionen för plattformen och de mål som hittats ska synas.
- En videoström från drönarens kamera där målen markeras genom information från bildbehandlingsmodulen.
- En lista över hittade mål.
- En ruta som visar plattformsinformation.
- En styrningsdel där användaren kan välja vilket uppdrag som ska utföras eller om drönaren ska styras manuellt.
- En terminal som kan användas vid utveckling samt för att skriva ut information.
- En knapp för nödstopp.

I tabell 4 ses vad som kommer skickas in till GUI:t och vad som GUI:t kommer skicka ut till övriga noder.

Tabell 4: GUI

In	Videoström Statusinformation för drönare Estimering av position för mål Osäkerhet för estimeringar Bildkoordinater för detekterade mål
Ut	Styrmod Styrkommandon Avsökningsområde

4.1 Gränssnitt

GUI:t körs med ROS-noden *gui*. Tabell 5 visar vilka topics som noden publicerar/prenumererar på. För att se vad en topic innefattar se avsnitt 3.

Tabell 5: Topics som noden gui prenumererar på och publicerar till.

	Topic
Subscribe	/goal_estimations /detection/image_coordinates /dji_sdk/local_position /dji_sdk/image_raw /dji_sdk/gps_health /dji_sdk/battery_state
Publish	/search_area /mission_state

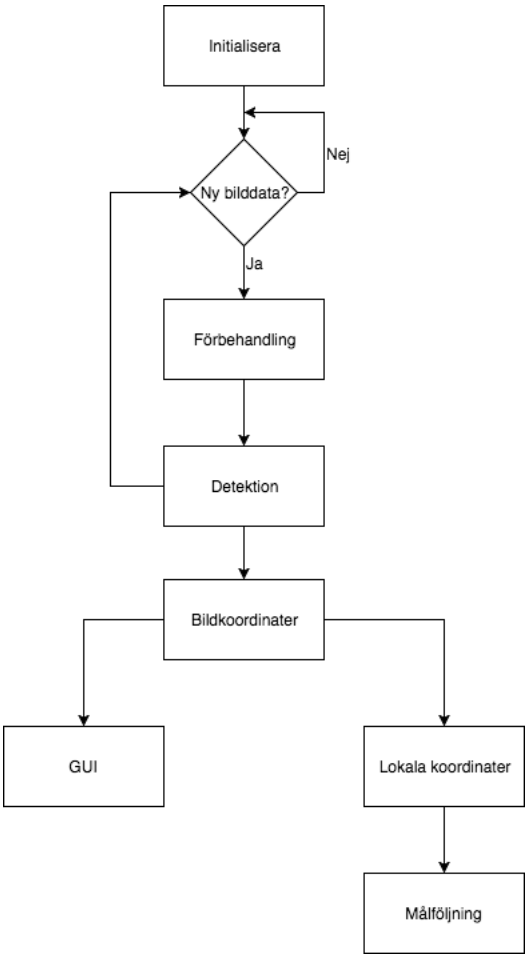
5 Bildbehandling

Bildbehandlingensmodulen är den modul som tar in live-videoströmmen från plattformen och behandlar datan. Bildbehandlingens uppgift är att detektera objekt och kartlägga dem. Bildbehandlingen ska utifrån videströmmen identifiera objekt och ta fram deras bildkoordinater. Efter detta ska bildkoordinaterna användas tillsammans med plattformens position för att ta fram objektens kartkoordinater.

Bildbehandlingen kommer implementeras med hjälp av programvarubiblioteket OpenCV [7].

5.1 Övergripande design

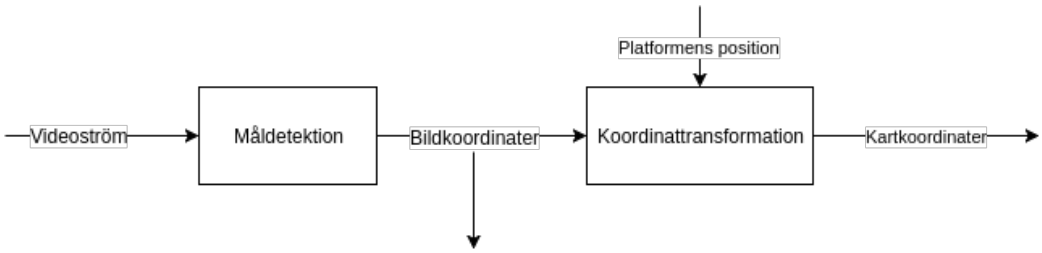
In- och utsignaler för modulen visas i tabell 6 och ett blockschema visas i figur 7. I figur 6 visas ett flödesschema över hur bildbehandlingsmodulen är tänkt att fungera.



Figur 6: Flödesschema för bildbehandlingssmodulen.

Tabell 6: Bildbehandling

In	Videoström Position för drönare
Ut	Kartkoordinater för detekterade mål Bildkoordinater för detekterade mål



Figur 7: Blockschema över bildbehandlingssmodulen.

5.2 Terminologi

Här beskrivs de uttryck och termer som används inom bildbehandlingsmodulen.

- RGB - Förkortning på röd, grön, blå vilka utgör grundfärgerna i additiv färgblandning.
- HSV - En färgrymd som består av beståndsdelarna hue, saturation, value.

5.3 Förbehandling

För att enklare kunna identifiera objekt i videoströmmen kommer den att förbehandlas med ett antal olika metoder. För att kunna leverera en produkt som fungerar så bra som möjligt kommer flera olika metoder för förbehandling att utvärderas. De metoder som kommer att analyseras är:

- Smoothing, genom att applicera ett filter på bilden fås en ”mjukare” bild med mindre brus. De två filter som kommer analyseras är ett Gaussiskt samt ett medelvärdesfilter. Kvaliteten på den kamera som kommer användas påverkar ifall videoströmmen är brusig eller inte och om smoothing kommer behövas.
- Histogram equalization, med hjälp av histogram över en bilds intensitet kan bildens kontrast förbättras vilket kan underlätta identifiering av objekt. Detta kan vara användbart när bilder har en bakgrund och förgrund så båda är antingen ljusa eller mörka.
- Downsampling, genom att filtrera bort högfrekventa komponenter i bilden kan bildbehandlingen utföras snabbare. Om de algoritmer som implementeras i bildbehandlingsmodulen är väldigt tidskrävande kommer downsampling att användas för att motverka detta.

- Morphological operations, det finns flera olika metoder för detta men de två som kommer användas är erode och dilate. Dessa metoder används för att minska bruset samt isolera individuella element i bilder. Dessa operationer kan vara användbara då de tar bort brus samtidigt som de isolerar objekt i bilder.

5.4 Detektera mål

För att detektera målen kommer färgbaserad detektion att användas. Nedan beskrivs teorin bakom att detektera objekt avseende på färg i en videoström.

Videoströmmen tar in bilder i RGB-format som sedan omvandlas till HSV-format, vilket utgör optimala färgrymden för färgbaserad bildsegmentering. Genom att ställa in trösklar för de olika representationerna kan en viss färg detekteras. Det krävs då att de färger som ska upptäckas definieras samt att tröskelvärdena för respektive färg identifieras.

Till exempel om röda bollar ska detekteras i en miljö av andra färger så kan trösklarna i HSV-formatet ställas för att extrahera alla färger förutom just rött. Utifrån detta är det enkelt att identifiera målen.

5.5 Markering av mål

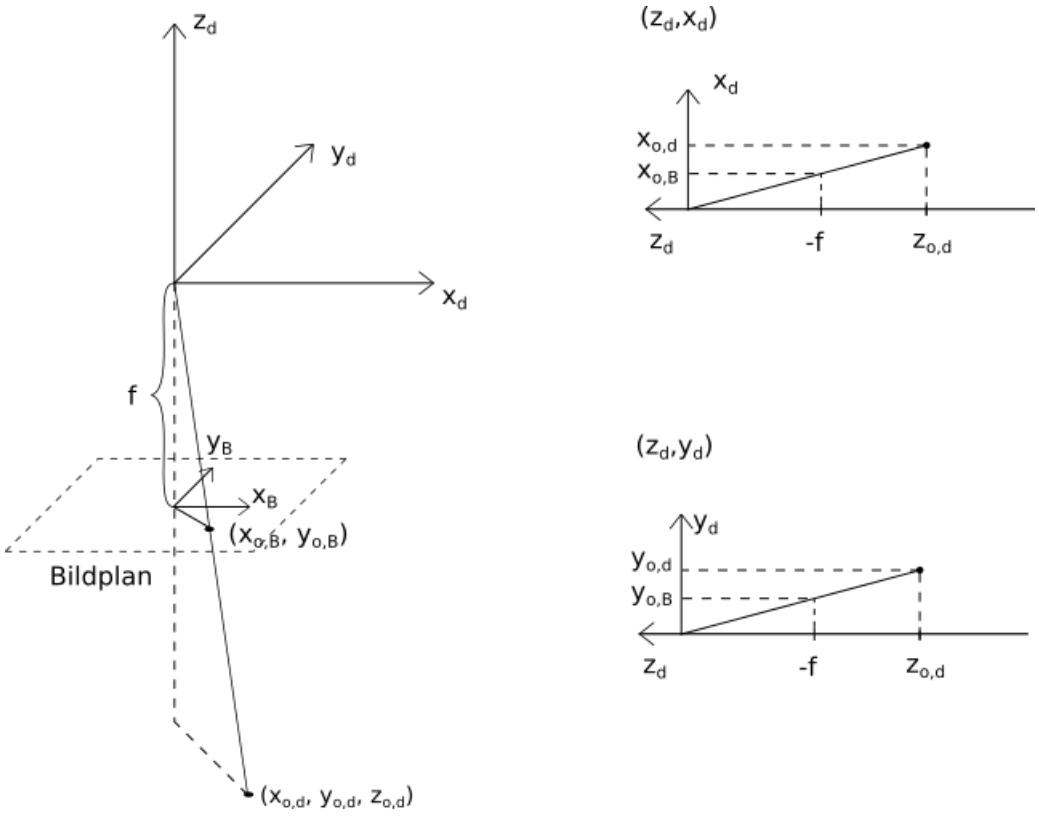
När ett mål har hittats ska bildbehandlingsmodulen ta fram koordinater och storlek för detta objekt, som talar om för GUI:t hur den ska rita ut markeringen. Först räknas målets mittpunkt ut och därefter tas sidolängderna fram för den rektangel som ska ritas ut runt objektet, oavsett objektets form.

5.6 Omvandling av koordinater

Bildbehandlingsmodulen kommer även vara ansvarig för att konvertera ett detekterat objekts koordinater i bilden till koordinater i ett lokalt system med origo positionerat vid drönarens startpunkt. Detta görs i två steg, först konverteras bildens tvådimensionella koordinater till tredimensionella koordinater i ett koordinatssystem med centrum i drönaren. Sedan konverteras punkten i drönarens koordinatsystem till det lokala systemet med hjälp av en rotations- och translationsmatris.

5.6.1 Konvertering av bildkoordinater

Konverteringen mellan objektets bildkoordinater och koordinater i drönarens koordinatsystem illustreras i figur 8.



Figur 8: Samband mellan drönarens och bildens koordinatsystem, till vänster är systemet beskrivet i 3D och till höger är figuren ritad i 2D sedd ur drönarens (z_d, x_d) respektive (z_d, y_d)-axlar.

Om den tredimensionella bilden ritas upp som (x,z) respektive (y,z) system fås två trianglar och ur dessa kan följande samband härledas:

$$\frac{-f}{z_{o,d}} = \frac{y_{o,B}}{y_{o,d}}, \quad \frac{-f}{z_{o,d}} = \frac{x_{o,B}}{x_{o,d}} \tag{1}$$

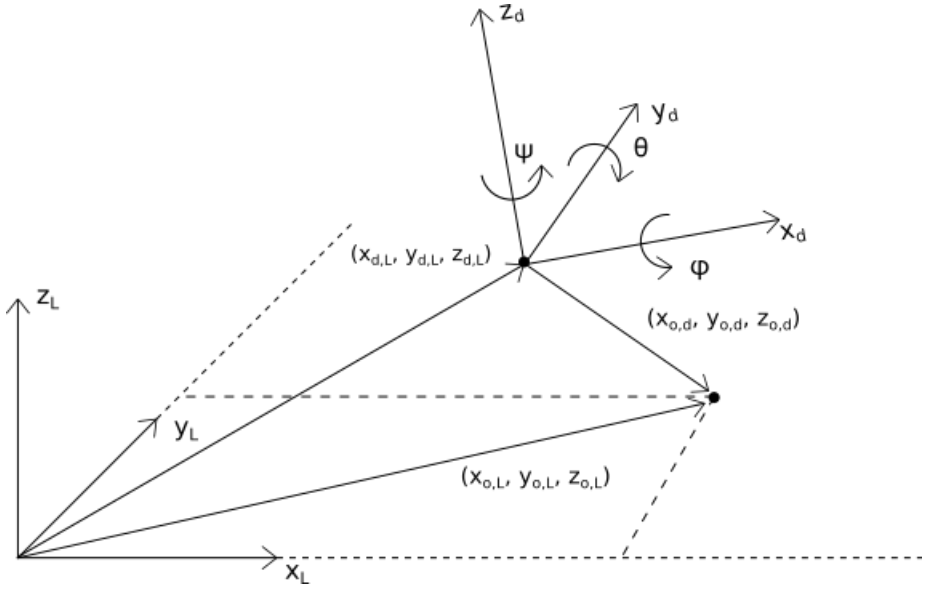
där ($x_{o,B}$,$y_{o,B}$) är objektets bildkoordinater, f är kamerans fokallängd och ($x_{o,d}$,$y_{o,d}$,$z_{o,d}$) är objektets koordinater i drönarens koordinatsystem. Då $z_{o,d}$ anger objektets position i höjd kommer denna antas vara samma som drönarens avstånd till marken (notera att detta värde kommer i drönarens koordinatsystem att vara negativt). Om ett uttryck för objektets position i drönarens koordinatsystem, $p_{o,d}$ löses ur ekvation 1 fås:

$$p_{o,d} = (x_{o,d}, y_{o,d}, z_{o,d}) = \left(\frac{z_{o,d}x_{o,B}}{-f}, \frac{z_{o,d}y_{o,B}}{-f}, z_{o,d}\right) \tag{2}$$

5.6.2 Konvertering till lokalt koordinatsystem

För att få ut de lokala koordinaterna för objektet, $(x_{o,L}, y_{o,L}, z_{o,L})$ behöver drönarens rotationer roll, tilt och gir samt drönarens position kompenseras för. När detta är gjort kommer drönarens koordinatsystem, $(\hat{x}_d, \hat{y}_d, \hat{z}_d)$, vara positionerat så att det har samma

orientering som det lokala koordinatssystemet, $(\hat{x}_L, \hat{y}_L, \hat{z}_L)$. En illustation av hur objektets position i det lokala koordinatsystemet beräknas med hjälp av drönarens position $(x_{d,L}, y_{d,L}, z_{d,L})$ och objektets position i drönarens koordinatsystem, $(x_{o,d}, y_{o,d}, z_{o,d})$ kan ses i figur 9. Konverteringen som detta resulterar i kan ses i ekvation 3 där rotations- och translationsmatrisen som åstadkommer detta ses i ekvation 4.



Figur 9: Relationen mellan objektets position i drönarens och objektets position i det lokala koordinatsystemet.

$$\begin{pmatrix} x_{o,L} \\ y_{o,L} \\ z_{o,L} \end{pmatrix} = R \begin{pmatrix} x_{o,d} \\ y_{o,d} \\ z_{o,d} \\ 1 \end{pmatrix}, \tag{3}$$

$$R = \begin{pmatrix} a_{11} & a_{12} & a_{13} & x_{d,L} \\ a_{21} & a_{22} & a_{23} & y_{d,L} \\ a_{31} & a_{32} & a_{33} & z_{d,L} \end{pmatrix}, \tag{4}$$

där

$$\begin{aligned} a_{11} &= \cos(\psi + \alpha)\cos(\theta), \\ a_{12} &= \sin(\psi + \alpha)\cos(\theta), \\ a_{13} &= -\sin(\theta), \\ a_{21} &= \cos(\psi + \alpha)\sin(\theta)\sin(\phi) - \sin(\psi + \alpha)\cos(\phi), \\ a_{22} &= \sin(\psi + \alpha)\sin(\theta)\sin(\phi) + \cos(\psi + \alpha)\cos(\phi), \\ a_{23} &= \cos(\theta)\sin(\phi), \\ a_{31} &= \cos(\psi + \alpha)\sin(\theta)\cos(\phi) + \sin(\psi + \alpha)\sin(\phi), \\ a_{32} &= \sin(\psi + \alpha)\sin(\theta)\cos(\phi) - \cos(\psi + \alpha)\sin(\phi), \\ a_{33} &= \cos(\theta)\cos(\phi), \end{aligned} \tag{5}$$

där ϕ är rollvinkeln, θ är tiltvinkeln, ψ är girvinkeln och α är en offset från när drönaren startas ifall den inte är roterad i liknande ψ-led som det lokala koordinatsystemet. Rotationerna utförs i ordningen $R_\phi^x R_\theta^y R_\psi^z$ där värdena $a_{i,j}$ är tagna från kursboken i sensorfusion [8]. Värdena $(x_{o,d}, y_{o,d}, z_{o,d})$ är objektets koordinater i drönarens koordinatsystem medan $(x_{d,L}, y_{d,L}, z_{d,L})$ är drönarens position i det lokala koordinatsystemet och används för translationen.

5.7 Gränssnitt

Bildbehandlingen utförs av ROS-noden *planning*. Tabell 14 visar vilka topics som noden publicera/prenumererar på. För att se vad en topic innefattar se avsnitt 3.

Tabell 7: Topics som noden image_processing prenumererar på och publicerar till.

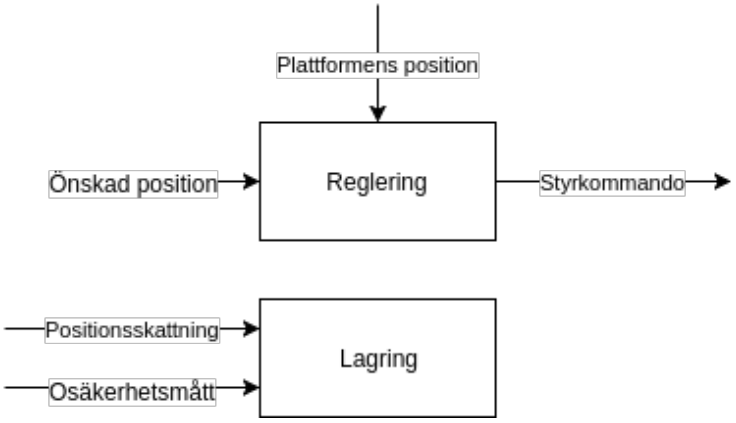
	Topic
Subscribe	/dji_sdk/image_raw /dji_sdk/local_position /dji_sdk/imu /dji_sdk/gps_health
Publish	/detection/map_coordinates /detection/image_coordinates

6 Positionering

Positioneringsmodulen kommer se till att drönaren följer den väg som uppdragsplaneraren tagit fram. Beroende på den mjukvara som finns på drönaren kommer positioneringsmodulen att skicka antingen GPS-koordinater eller styrkommandon för att åstadkomma detta. Positoneringsmodulen kommer även att lagra drönarens position och skattningar av målens position. I tabell 8 ses in- och utsignaler till modulen. Figur 10 visar ett blockschema över modulen.

Tabell 8: Positionering

In	Trajektoria/önskad position Drönarens GPS-koordinater Skattningar av målens position
Ut	Styrkommandon



Figur 10: Blockschema över positioneringsmodulen.

6.1 Gränssnitt

Positioneringen utförs av ROS-noden *positioning*. Tabell 9 visar vilka topics som noden publicera/prenumererar på. För att se vad en topic innefattar se avsnitt 3.

Tabell 9: Topics som noden positioning prenumererar på och publicerar till.

	Topic
Subscribe	/goal_estimations /planned_trajectory /dji_sdk/local_position /dji_sdk/gps_health

Styrkommandon till plattformen ges i form av waypoints och dessa skickas via services. Vilka services som noden begär visas i tabell 10.

Tabell 10: Services som noden positioning begär.

Service	**Provider**
/dji_sdk/mission_waypoint_upload	dji_sdk
/dji_sdk/mission_waypoint_action	dji_sdk
/dji_sdk/sdk_control_authority	dji_sdk

7 Målföljning

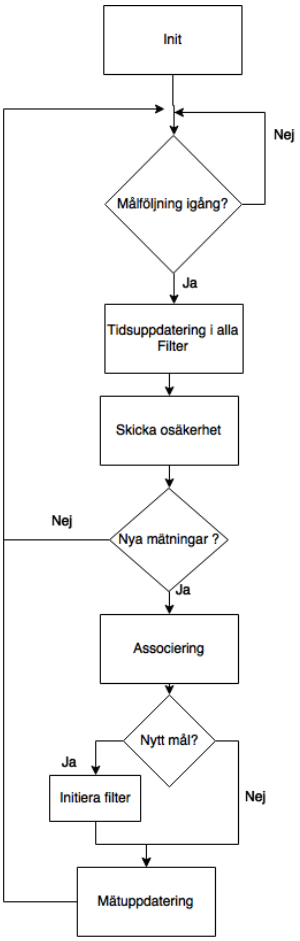
Modulens uppgift är att skatta positioner för mål och uppskatta hur osäkra dessa är. Skattningarna baseras på kartkoordinater för detekterade mål som erhålls från bildbehandlingen. I det fallet kommer kartkoordinaterna inte att vara associerade med ett specifikt mål vilket innebär att en sådan association måste göras innan skattningarna kan uppdateras. Associationen görs genom att hitta det mål vars senaste skattning har minst avstånd till den aktuella detektionen.

Eventuellt kommer gruppen även använda blåtandtaggar som fästs på målen för att lokalisera och identifiera dem.

Projektgruppen kommer att använda ett linjärt Kalmanfilter med en konstant positionsmodell (CP-model) för att tracka målen. Filtret kommer bara att utföra en mätuppdatering om målet har detekterats.

7.1 Övergripande design

I figur 11 visas ett flödesschema för målföljningsmodulen.



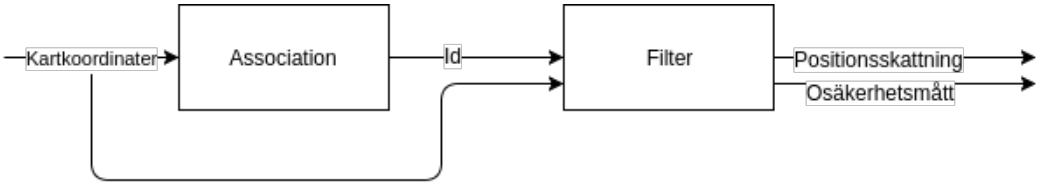
Figur 11: Flödesschema för målföljningssmodulen.

I tabell 11 visas in- och utsignaler för modulen.

Tabell 11: Målföljning

In	Kartkoordinater för detekterade mål
Ut	Estimering av position för mål Osäkerhet för estimering

I figur 12 visas ett blockschema över målföljningsmodulen.



Figur 12: Blockschema över målföljningsmodulen.

7.2 Rörelsemodell

Målens positioner antas vara konstanta och därför används en konstant positionsmodell (CP-model). Position i x-led respektive y-led utgör målets tillstånd. Tillståndsvektorn blir då enligt följande,

$$x = \begin{pmatrix} x_L \\ y_L \end{pmatrix} \tag{6}$$

där x_L och y_L är givna i det globala kartesiska koordinatsystemet. Mätekvationen är linjär eftersom mätningarna som fås in beskrivs av samma globala koordinatsystem som tillståndsvektorn. Således beskrivs tillståndsmodellen av

$$x_{k+1} = Fx_k + G_v v_k \quad cov(v_k) = Q \tag{7}$$
$$y_k = Hx_k + e_k \quad cov(e_k) = R \tag{8}$$

där F, G_v och H ges av

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, G_v = T_s \begin{bmatrix} 1 \\ 1 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

I ekvationen ovan är T_s samplingstiden. Kovariansen för modellbruset (Q) och kovariansen för mätbruset (R) kommer justeras till lämpliga värden under projektets gång. De antas vara på formen

$$Q = \sigma_Q^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \sigma_R^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

7.3 Tidsuppdatering

Tidsuppdateringen ges av ekvation 9 [8].

$$\begin{aligned} \hat{x}_{k+1|k} &= F\hat{x}_{k|k} \\ P_{k+1|k} &= FP_{k|k}F^T + G_v Q G_v^T \end{aligned} \tag{9}$$

7.4 Associering

Ett problem som uppstår i målföljningsmodulen är att associera inkommande mätdata med information som redan finns. Modulen ska kunna avgöra om ett detekterat objekt är ett helt nytt eller om det är ett tidigare detekterat objekt där en mätuppdatering behöver göras. För att kunna göra detta måste man beräkna avståndet ett objekt kan röra på sig under en viss tidpunkt.

7.4.1 Avståndsberäkning

För att kunna göra en avståndsberäkning på hur långt ett mål har rört sig under en viss tid används Gating. En Gate formas runt målets predikterade position och tar formen av en elliptisk kurva. Gate:en räknas ut enligt följande, där S_k^{-1} ges av ekvation 12 nedan.

$$(y_k - \hat{y}_{k|k-1})^T S_k^{-1} (y_k - \hat{y}_{k|k-1}) < G \tag{10}$$

Vänsterledet i ekvation 10 är χ^2 fördelad med två frihetsgrader då mätdatan har två dimensioner. G väljs efter vilken sannolikhet vi tillåter att ett mål som är associerat med en gate, befinner sig utanför gate:en. Tillexempel om vi vill ha en säkerhet på 95% att ett mål befinner sig inom gaten väljer vi G till $\chi^2(0.95) = 5.99$.

7.4.2 Nearest Neighbor

För att kunna avgöra vilket objekt som tillhör vilka koordinater används en variant av algoritmen Nearest neighbor. Algoritmen bygger på att för varje gate antar man att det mål som ligger närmast den predikterade positionen är det mest sannolika. Mätdatan mappas enligt följande ekvation.

$$z = \underset{1 \leq i \leq m_k}{\arg\min} (y_k^i - \hat{y}_{k|k-1}^i)^T S_k^{-1} (y_k^i - \hat{y}_{k|k-1}^i) \tag{11}$$

Där m_k är antalet mål inom gaten. Om flera objekt finns detekterade observerar algoritmen vilket mål inom gaten som ligger närmast den predikterade positionen och associerar dessa. Ett mål kan inte associeras fler än en gång. Om ett mål detekteras och plattformen inte befinner sig inom någon gate kommer den registreras som ett nytt mål.

Nearest neighbor är en av de enklare algoritmer för detta ändamål, trots det anses den ge tillräckliga resultat. Om det inte är fallet och om tid finns tillgänglig kommer ett försök att implementera en variant av Hungarian algorithm göras. Kortfattat sätter den upp en kostnadsmatris och minimerar kostnaden för alla detekterade mål.

7.5 Mätuppdatering

Mätuppdateringen för Kalmanfiltret beskrivs av ekvation 12 nedan [8].

$$\begin{aligned} S_k &= R + HP_{k|k-1}H^T \\ K_k &= P_{k|k-1}H^T S_k^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k\epsilon_k \\ P_{k|k} &= (I - K_k H)P_{k|k-1} \end{aligned} \tag{12}$$

7.6 Gränssnitt

Målföljningen utförs av ROS-noden *tracking*. Tabell 12 visar vilka topics som noden publicera/prenumererar på. För att se vad en topic innefattar se avsnitt 3.

Tabell 12: Topics som noden tracking prenumererar på och publicerar till.

	Topic
Subscribe	/detection/map_coordinates
Publish	/goal_estimations

8 Uppdragsplanering

Modulens uppgift är att utifrån ett givet uppdrag se till att en trajektoria beräknas för att uppfylla uppdraget.

Dessa uppdrag innefattar:

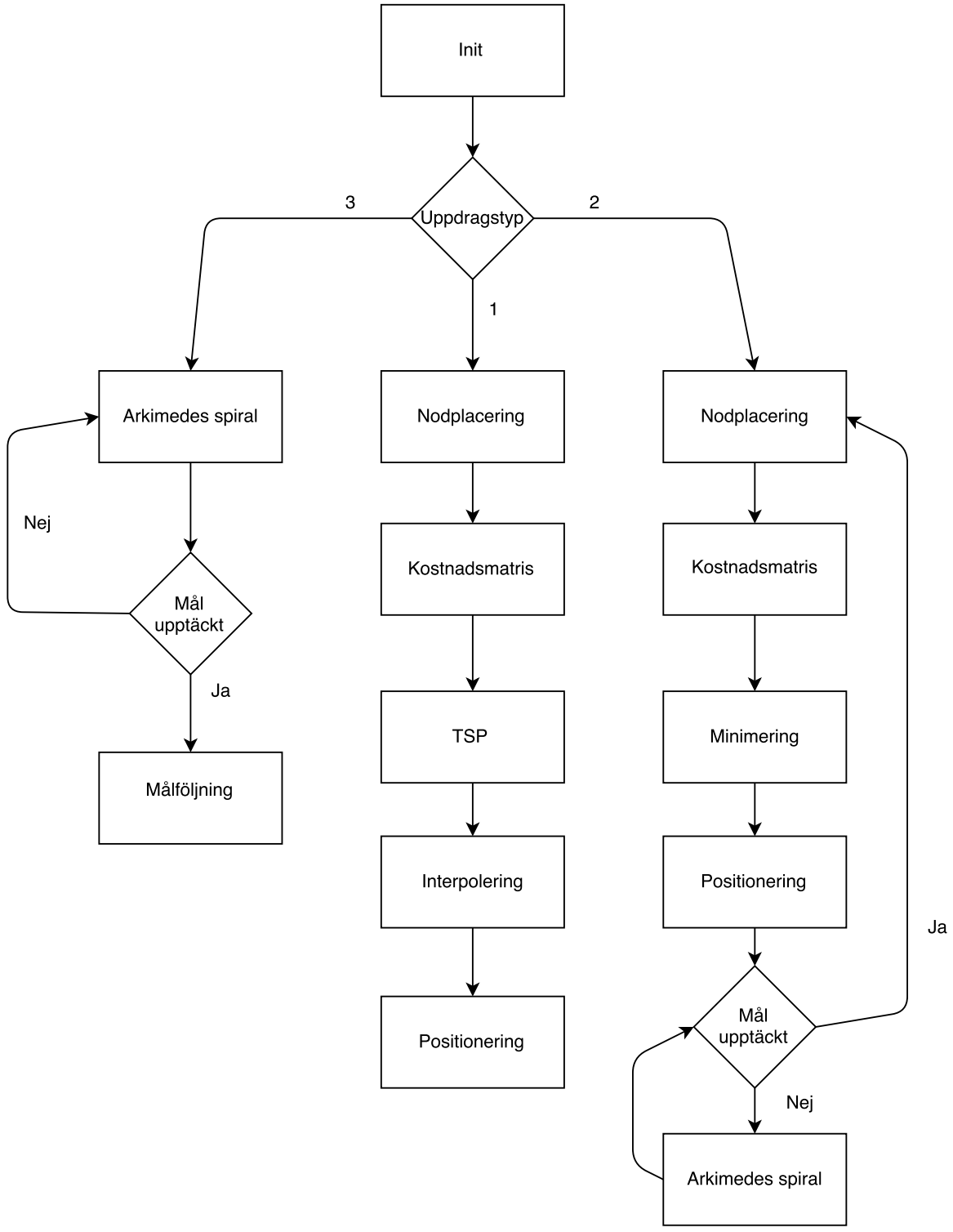
1. Plattformen ska kunna söka av ett definierat område och identifiera målen samt spara deras positioner.
2. Plattformen ska kunna kontinuerligt estimera positionen på ett fördefinierat antal mål.
3. Plattformen ska utifrån en specificerad position kunna identifiera det närmsta målet och följa detta.

8.1 Övergripande design

I tabell 13 visas in- och utsignaler för modulen och i figur 13 visas ett flödesschema för modulen.

Tabell 13: Uppdragsplanering

In	Estimering av position för mål Osäkerhet för estimeringar Position för drönare Styrmod Avsökningsområde
Ut	Trajektoria/önskad position



Figur 13: Flödesschema för uppdragsplanering.

8.2 Uppdrag 1

Det första uppdraget listat ovan innebär att plattformen ska kunna söka av ett definierat område, identifiera eventuella mål, samt spara deras positioner. Det specificerade sökområdet delas in i ett antal noder, vilket görs i blocket *Nodplacering* i figur 13. För att minimera flygtiden och effektivisera arbetet ställs uppdragsplanering då inför ett handelsresandeproblem.

Detta löses med lämplig heuristik i blocket *TSP*, där TSP står för Traveling Salesman Problem. *Lin-Kernighans heuristik* (LKH) kommer användas, så länge inga problem uppstår.

LKH är i dagsläget en av de effektivare algoritmer för att lösa TSP problem. LKH-algoritmen utgår från en godtycklig väg mellan noderna, sedan letar den efter ett set av kortare/billigare bågar som kan byta ut ett set av de nuvarande bågar. Processen itereras tills inga kortare set med bågar hittas. Antalet bågar i dessa sätts vanligtvis till två eller tre stycken. [9]

Bågkostnaderna sätts i blocket *Kostnadsmatris* till avstånden mellan noderna, eventuella förbjudna områden behandlas genom att skarpt bestraffa bågarna som går genom dessa områden. Innan den resulterande trajektorian skickas vidare till positioneringsmodulen görs en interpolering av trajektorian för att undvika skarpa kurvor, mest för plattformens skull men också för att underlätta identifiering av eventuella mål. Om trajektorian efter interpolering går genom ett förbjudet område förkastas den vägen. Istället görs en skarp kurva för att undvika det förbjudna området. Att plattformens trajektoria inte får korsa förbjudna områden är ett krav med prioritet 2 och därför prioriteras i första hand bara att plattformen ska hålla sig innanför det givna sökområdet.

8.3 Uppdrag 2

Vid kontinuerlig estimering av målpositioner kommer en önskad position $\hat{p}$ beräknas för plattformen genom en planeringsalgoritm på formen

$$\hat{p} = \underset{p \in \{x_1,..,x_n\}}{\arg\min} \left(\alpha \cdot \sum_i q_i ||\hat{x}_i - p|| + (1-\alpha) \cdot ||p - p_0|| \right) \quad , \quad i \notin i_s \ , \quad \alpha \in [0,1] \qquad (13)$$

där $\hat{x}_i$ är senaste positionsskattningen för mål i, q_i är ett osäkerhetsmått på denna skattning, p_0 är plattformens nuvarande position och i_s är det senaste besökta målet.

Initialt väljs kostnadsfunktionen till en linjär viktning mellan osäkerheten på målens positionsskattning och avstånden till positionskattningarna, med viktningsfaktorn α. I mån av tid kan en mer avancerad algoritm användas för att minimera flygsträckan och därmed den totala osäkerheten för positionsskattningarna. Om målen inte hittas vid sin senaste estimerade position, används Arkimedes spiral för att söka av närområdet tills målet hittas. När målet hittats kommer ett nytt $\hat{p}$ räknas ut, det funna målet kommer då uteslutas ur ekvationen och det senaste målet (i_s) sätts åter in i mängden i.

8.4 Uppdrag 3

I uppdrag tre ska plattformen identifiera och följa det närmsta målet utifrån en specificerad position. Detta problem löses med Arkimedes spiral. Arkimedes spiral utgår ifrån en fix punkt (i det här fallet plattformens specificerade position) och rör sig sedan med konstant hastighet bort från punkten utefter en linje. Linjen roterar samtidigt med konstant vinkelhastighet runt den fixa punkten. Hastighet samt vinkelhastighet anpassas efter kamerans täckningsyta för att minska möjligheten att ett mål förbises.

Plattformen rör sig sedan längs spiralen tills ett mål upptäcks. Då avbryts rörelsen längs spiralen och målet följs istället. Ekvationen för Arkimedes spiral ges här nedanför

$$x = (a_x + b \cdot \theta) \cdot \cos(\theta)$$
$$y = (a_y + b \cdot \theta) \cdot \sin(\theta)$$

Där **a** är centrumkoordinaten, b är avstånd mellan spiralerna och θ är vinkeln, $\theta \in [0, 2\pi n]$ där n är antalet varv.

8.5 Gränssnitt

Uppdragsplaneringen utförs av ROS-noden *planning*. Tabell 14 visar vilka topics som noden publicerar/prenumererar på. För att se vad en topic innefattar se avsnitt 3.

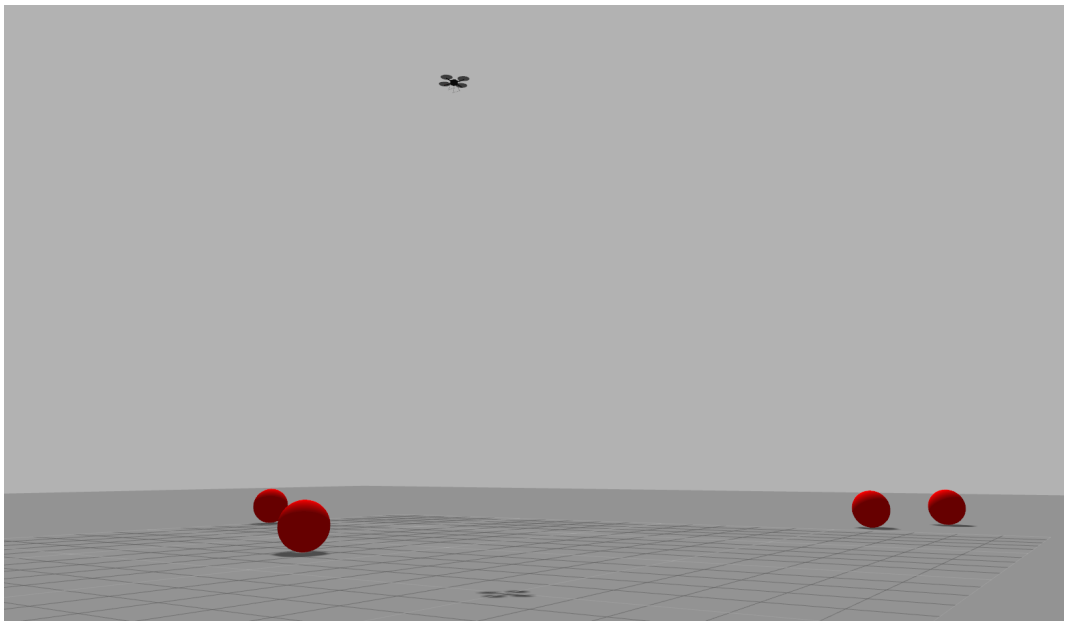
Tabell 14: Topics som noden planning prenumererar på och publicerar till

	Topic
Subscribe	/goal_estimations /search_area /mission_state /dji_sdk/local_position /dji_sdk/gps_health
Publish	/planned_trajectory

9 Simulering

En simuleringsmiljö i Gazebo[10] kommer att upprättas för att kunna simulera systemet och ska användas för att verifiera att varje modul uppfyller sina krav.

Simuleringsmiljön kommer att bestå av en plattform som kan flyga till en given kartkoordinat och vara utrustad med en nedåtriktad kamera, en IMU samt en GPS-mottagare. I simuleringen kommer det finnas rörliga mål på marken som ska användas för att utvärdera modulernas funktion och integration. En figur från simuleringsmiljön med en plattform och fyra mål visas i figur 14. Reglering av motorer och dylikt kommer inte simuleras eftersom plattformen antas ha funktionalitet att flyga till en given kartkoordinat (finns i DJI:s Onbard SDK [11]).



Figur 14: Exempelbild från simuleringsmiljön i Gazebo. De röda sfärerna representerar mål.

I simuleringen kommer en ”generell” plattform att användas, alltså inte specifikt DJI Matrice 100, då det i nuläget är oklart när denna levereras. Den simulerade plattformen kommer att emulera kommunikationen från en DJI Matrice 100 som är kopplad till en dator med ros-paketet dji_sdk [3]. Med detta menas att den simulerade plattformen kommer att publicera och prenumerara på samma topics som noden dji_sdk kommer att publicera och prenumerera på.

Referenser

[1] *std_msgs - ROS Wiki*, http://wiki.ros.org/std_msgs, Hämtad 2017-09-26.

[2] *common_msgs - ROS Wiki*, http://wiki.ros.org/common_msgs, Hämtad 2017-09-26.

[3] *dji_sdk ROS Wiki*, http://wiki.ros.org/dji_sdk, Hämtad 2017-09-24.

[4] *Qt Documentation*, http://doc.qt.io, Hämtad 2017-09-18.

[5] *Marble*, https://marble.kde.org/, Hämtad 2017-09-25.

[6] *rqt - ROS Wiki*, http://wiki.ros.org/rqt, Hämtad 2017-09-25.

[7] *About - OpenCV library*, http://opencv.org/about.html, Hämtad 2017-09-18.

[8] F. Gustafsson, *Statistical Sensor Fusion*. Studentlitteratur, 2012.

[9] *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.1798&rep=rep1&type=pdf, Hämtad 2017-10-03.

[10] *Gazebo*, http://gazebosim.org, Hämtad 2017-09-18.

[11] *DJI SDK ROS Map Waypoint Navigation Package*, https://developer.dji.com/onboard-sdk/documentation/github-platform-docs/ROS_Example/ros_map_waypoint_navigation_package.html, Hämtad 2017-09-18.