KU Leuven
Departement Computerwetenschappen

P&O: Computerwetenschappen

Eindverslag

Team:
Zilver

Bram Vandendriessche (Coördinator)
Arne Vlietinck (Secretaris)
Matthias Van der Heyden
Jef Versyck
Vincent Vliegen
Laura Vranken

Academiejaar 2016-2017

Samenvatting

Auteur:

Inhoudsopgave

Inleiding

Auteurs: Laura Vranken & Arne Vlietinck

Drones zijn de laatste jaren enorm in populariteit toegenomen en blijven hierdoor ook in positieve zin evolueren. Ze worden tegenwoordig gebruikt voor talloze toepassingen. De bekendste toepassing bevindt zich binnen Defensie, die drones gebruiken om informatie te verkrijgen over vijandelijk gebied zonder mensenlevens te moeten riskeren. Daarnaast hebben ook grote bedrijven (o.a. Amazon[1]) de weg naar deze technologie gevonden. De toekomst brengt echter nog veel meer voordelen. Enkele voorbeelden zijn veiligheidsinspectie van windturbines of elektriciteitslijnen, luchtsteun bij zoek- en reddingsoperaties, bewaking en luchtfotografie. [**?**]
Wanneer een drone autonoom functioneert, is een betrouwbare aansturing door de Autopilot van levensbelang. Hij moet namelijk bestand zijn tegen allerlei externe factoren (bv. wind).

Dit verslag behandelt de autonome aansturing van een drone, meer bepaald een quadcopter en is een vervolg op het tussentijds verslag. [**?**] Er wordt uitgegaan van een drone waarop twee voorwaarts gerichte camera's bevestigd zijn. Op basis van deze beelden moeten afstand en positie van het doel ingeschat worden en nieuwe bewegingsopdrachten voor de drone gegenereerd worden. Deze bewegingen worden weergegeven in een Virtual Testbed. Dit is een softwaresysteem dat een fysieke opstelling van een drone en camera's simuleert. [**?**] De simulator genereert beelden van de drone in verschillende standpunten a.d.h.v. de verkregen bewegingsopdrachten van de Autopilot.

De Autopilot en Virtual Testbed moeten zo ontworpen worden dat de drone in staat is om zijn doel, een niet grijze bol, te lokaliseren en ernaar toe te vliegen. Dit eventueel onder lichte invloed van wind in willekeurige richtingen. Bovendien moet ook voor beiden een grafische user interface (*GUI*) ontworpen worden. De *GUI* toont de vooruitgang en laat de gebruiker toe allerlei informatie (snelheid, positie en verschillende camerastandpunten) op te vragen. Daarnaast kan de gebruiker nieuwe bollen toevoegen en de wind manueel aanpassen in de verschillende richtingen.

De tekst is als volgt opgebouwd.

1 Ontwerp

Auteur: ; redactie: Arne Vlietinck

1.1 Drone Autopilot

Auteur: Laura Vranken

De Drone Autopilot bepaalt de positie van de drone relatief ten opzichte van zijn doel a.d.h.v. twee beelden gegeneerd door de dronecamera's. Bovendien zorgt de Autopilot ervoor dat de drone juist naar zijn doel toe vliegt. De drone bereikt zijn doel wanneer hij door een gekleurde bol gevlogen is of door ze allemaal in geval van meerdere bollen. Daarnaast moet de Autopilot ook rekening houden met een mogelijke invloed van wind die de drone van zijn koers doet afwijken als ook met obstakels waarrond hij moet vliegen.

1.1.1 Beeldverwerking

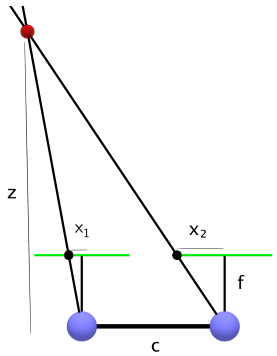
Ten eerste moeten de beelden die de Autopilot van de Virtual Testbed binnenkrijgt, geanalyseerd worden. Dit gebeurt door iteratief de kleurwaarden van elke pixel te vergelijken met de waarde van de opgegeven kleur indien er al een doel beslist is. Anders zullen de pixels gegroepeerd worden per kleur dat voorkomt in een HashMap. De gekleurde pixels worden bijgehouden door hun positie ten opzichte van het beeld, uitgedrukt in rij en kolom, op te slaan. We baseren onze berekeningen op het midden van de bol. Dit kan benaderd worden op twee manieren: via het zwaartepunt of de kleinstekwadratenmethode op de randpunten van de cirkel. Het zwaartepunt van een bepaalde kleur pixels

[1] Amazon Prime Air

is te berekenen via het gemiddelde van de opgeslagen coördinaten. De kleinste-kwadratenmethode zoekt daarentegen eerst de randpunten uit van de cirkel. Deze worden vervolgens gebruikt in een algoritme, dat de cirkel bepaalt die het beste past in de gegeven randpunten. Hieruit kan dan de positie van het centrum van de bol bepaald worden. [**?**] De Autopilot zal eerst gebruik maken van de kleinste-kwadratenmethode en overschakelen op de zwaartepuntberekening wanneer er onvoldoende randpunten zijn, aangezien deze minder nauwkeurig is wanneer het middelpunt buiten beeld ligt. TODO: Vincent moet bij algoritmen zijn kleinste kwadraten meer uitleggen!
Indien de Autopilot geen gekleurde pixels detecteert, zal de drone geleidelijkaan de wereld afscannen. Hierover meer info in subsectie 1.1.3.

1.1.2 Vliegstrategie

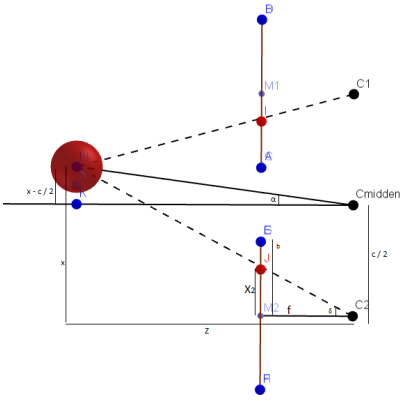
Wanneer de Autopilot iets in beeld gekregen heeft, zal hij starten met zijn positie relatief tegenover zijn doel te bepalen. Dit gebeurt in volgende stappen.
Ten eerst wordt de diepte bepaald. Dit kan met behulp van de formule van stereo vision [**?**] uitgewerkt worden. Zie Figuur 1 voor een grafische weergave van de berekening. Z stelt de diepte [m] voor, c de afstand [m] tussen de camera's, f de focale afstand [pixel] en x_1 en x_2 stellen de afstand [pixel] voor tussen het middelpunt van het beeld en het middelpunt van de bol op het beeld voor.



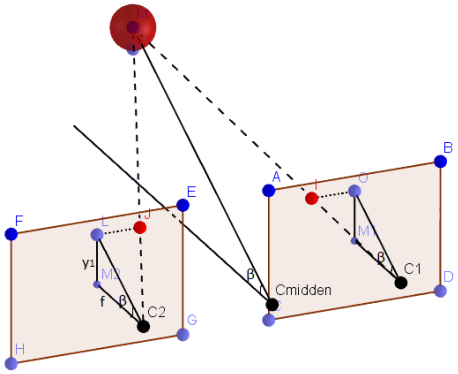
Figuur 1: Diepteberekening tussen drone en doel. In formulevorm: $Z = \frac{c*f}{x_1 - x_2}$.

Vervolgens bepalen we de fout op de horizontale hoek, α. Hiermee wordt de afwijking tussen de horizontale positie van de bol en het midden van de drone bedoeld. Deze formule wordt afgeleid via de goniometrische regels. Zie Figuur 2 voor grafische ondersteuning. De hoek δ stelt hier de helft van de horizontale hoek die het beeld overspant, voor en b stelt de helft van de breedte van het beeld voor. Onder de figuur, kan eveneens ook de uitwerking van de formule gevonden worden.
Tenslotte bepalen we ook de fout op de verticale hoek, β. Dit is de afwijking van de hoogte van de bol ten opzichte van de hoogte van de drone. Wederom afgeleid via de goniometrie en weergegeven in Figuur 3, met y_2 de verticale afstand [pixel] tussen de het middelpunt van het beeld en het middelpunt van de bol.
Nadat de drone zijn positie bepaald heeft, kan hij zijn fouten (horizontale en verticale hoek en eventuele roll) bijsturen a.d.h.v. PI controllers die beslissen over respectievelijk yaw, thrust en roll bewegingen. Meer info over de werking van de controllers wordt gegeven in subsectie 1.1.4. Deze bijsturing gebeurt tijdens het vliegen naar het doel.

Nu rest er ons enkel nog de pitch te bepalen zodat we onder een ingestelde thrust voorwaarts richting de bol kunnen bewegen. Het eerste idee was om dit op basis van een soort van snelheidscontroller te doen, aangezien de snelheid afhankelijk is van de pitch en dat we hierdoor ook de snelheid konden controlleren en laag houden. De snelheid is echter niet te bepalen door de onbekende windkrachten of door de numeriek wiskundige beperkingen op afgeleiden berekenen. Bijgevolg zijn we overgegaan op het tweede plan. Dit plan houdt in dat de drone pitcht met een rate op basis van een afstandscontroller, zodat hij een kleine afstand overbrugt, terug recht komt wat de snelheid een klein beetje afbouwt en dan weer opnieuw pitcht om verder te gaan. Wanneer er tegenwind is, zal de afstandscontroller de drone toelaten om verder te pitchen om toch op zijn



Figuur 2: Relatieve horizontale hoek.



Figuur 3: Relatieve verticale hoek.

In Figuur 2 wordt de relatieve horizontale hoek tussen drone en doel weergegeven door $\tan(\alpha) = \frac{x-\frac{c}{2}}{Z}$, voor de volledige uitwerking zie formule 1 tot 4.
In Figuur 3 wordt de relatieve verticale hoek weergegeven tussen drone en doel. De relatie wordt gegeven door volgende formule: $\tan(\beta) = \frac{y_1}{f}$.

Berekening brandpuntsafstand f:

$$\tan(\delta) = \frac{b}{f} \quad (1)$$

$$f = \frac{b}{\tan(\delta)} \quad (2)$$

Berekening afstand x:

$$\frac{x_2}{f} = \frac{x}{z} \quad (3)$$

$$x = z * \frac{x_2}{f} \quad (4)$$

doel te raken.

Tenslotte moet dit proces herhaaldelijk worden uitgevoerd ten gevolge van de invloed van wind. De wind kan de drone namelijk uit koers brengen. Hierdoor zal de drone telkens zijn positie moeten herberekenen en zich opnieuw oriënteren en de controllers laten bijsturen.
De bijkomstige moeilijkheid van obstakels zal verder worden uitgewerkt in subsectie 1.1.5.

1.1.3 Scannen wereld

Wanneer de drone geen bollen meer in beeld heeft, wordt verwacht dat hij rond zich begint te zoeken totdat hij een nieuwe bol waarneemt. Om de hele wereld te kunnen afscannen, voeren we volgende stappen uit.

Ten eerste begint de drone rond zijn as te draaien d.m.v. een yaw beweging. Hij krijgt de opdracht dit te doen gedurende 540°. We nemen een overschot van 180°zodat hij zeker volledig rondgedraaid heeft, mits invloed van eventuele rotationele wind.
Vervolgens vliegt de drone achteruit voor een bepaalde afstand zodat mogelijke bollen die zich boven of onder de drone bevinden nu ook zichtbaar worden. Dit gebeurt door eerst omhoog te pitchen en achterwaartse snelheid op te bouwen, om vervolgens weer horizontaal te komen zodat ook de bollen onderaan zichtbaar blijven. We maken van de opgebouwde snelheid in het begin gebruik om ver genoeg achteruit te drijven.

Indien we na dit proces nog steeds niks waargenomen hebben, herstarten we dit proces.

1.1.4 PI Controllers

Om een PI controller te kunnen gebruiken, is het belangrijk om te weten wat het is en wat het doet. Volgens [**?**] fungeert de controller als een controlelus-feedback mechanisme. Dit systeem

gaat proberen de fout op de meetwaarde proberen te corrigeren. Het berekent namelijk continu de foutwaarde e(t) als het verschil tussen de verlangde waarde en de gemeten waarde. Op basis van deze gegevens berekent de controller de nodige correctie.

Er is gekozen voor een PI controller i.p.v. een PID controller, aangezien die ook in de meeste reële systemen wordt gebruikt. Bovendien is de term D (Derivative) heel gevoelig aan ruis.
PI staat voor Proportional-Integral. Het P-deel zorgt ervoor dat het verschil in verlangde waarde en de gemeten waarde met een factor K_p wordt versterkt. I zorgt voor een constante sommatie van de fout en vergroot de correctiewaarde afhankelijk van hoelang er een fout bestaat tussen gemeten en verlangde waarde. Dit wordt met een factor K_i versterkt. Deze constanten worden manueel bepaald door de reactie van de controller uit te plotten in grafieken en hun reacties te optimaliseren door deze coëfficiënten bij te sturen.

Tenslotte wordt de correctie dan op basis van de volgende formule berekend.

$$u(t) = K_p * e(t) + K_i * \int e(t)dt \tag{5}$$

Hier is u(t) de correctie, K_p de proportional constante, K_i de integral constante en e(t) de foutwaarde.

Nu gaan we deze theorie toepassen op ons eigen project.
Ten eerste bekijken we de roll controller. Hij heeft als doel de roll gelijk aan nul te houden. Dit is dan ook ineens de verlangde waarde. De controller begint slechts bij te sturen wanneer de roll groter is dan 0,2°in positieve of negatieve zin. Bijsturen gebeurt door de correctiewaarde als input te nemen voor de roll rate.
Vervolgens hebben we de yaw controller. Deze werkt enkel wanneer de doel bol in beeld is, aangezien hij een oriëntatiepunt nodig heeft om de yaw te kunnen bepalen. De verlangde waarde is gelijk aan een horizontale afwijking α van nul graden. De controller gaat dit bijsturen op dezelfde manier als bij de roll controller.
De hoogte controller stelt de thrust waarde in en verschilt licht van de anderen doordat hij de correctiewaarde optelt bij de standaard waarde om de zwaartekracht tegen te gaan. Verder probeert hij ook weer de verticale afwijking naar nul te brengen.
De pitch controller wordt gebruikt om zo snel mogelijk te kunnen gaan naar hover en dus een pitch van nul te krijgen. Wederom op dezelfde manier.
Tenslotte is er ook nog een afstandscontroller. Deze gaat proberen de afstand naar nul te reduceren en helpt bij het voorwaarts vliegen om de pitchrate in te stellen, zoals eerder uitgelegd.

1.1.5 Obstakels

De laatste hindernis is het ontwijken van obstakels. De eerste taak is het bepalen van de positie van de obstakels. Dit gebeurt door eerst de verschillende tinten grijs uit het beeld te filteren en vervolgens de obstakels te verwijderen die niet in de buurt van onze vliegrichting liggen. Tenslotte kiezen we de dichtste eruit, aangezien we die eerst zullen moeten ontwijken.

Het ontwijken zelf gebeurt a.d.h.v. de hoogte controller. Indien de bol in de bovenste helft van het beeld staat, zullen we onder de bol doorvliegen. Indien de bol in de onderste helft van het beeld staat, zullen we erover vliegen. We hebben gekozen om er onder en boven te vliegen aangezien de reactie op de thrust vrij snel is. De andere optie is er naast te vliegen. Dit zou veel moeilijker zijn omdat de drone nog voorwaarts zal verder drijven, ook al is de yaw aangepast. De roll aanpassen om dat probleem op te lossen, zou veel moeilijker zijn. Vandaar de keuze om de hoogte aan te passen.

1.2 Virtual Testbed

Auteur:

2 Algoritmes

Auteurs: ; redactie: Arne Vlietinck

2.1 Drone Autopilot

Auteurs: Matthias Van der Heyden, Laura Vranken, Vincent Vliegen & Arne Vlietinck

2.2 Virtual Testbed

Auteur: Jef Versyck

3 Software

Auteurs: ; Redactie: Arne Vlietinck

3.1 Drone Autopilot

Auteur: Laura Vranken

3.2 Virtual Testbed

Auteur: Bram Vandendriessche

4 GUI

Auteurs: ; redactie: Arne Vlietinck

4.1 Drone Autopilot

Auteur: Matthias Van der Heyden

4.2 Virtual Testbed

Auteur: Arne Vlietinck

5 Testen

Auteurs: ; redactie: Arne Vlietinck

5.1 Drone Autopilot

Auteur: Jef Versyck

5.2 Virtual Testbed

Auteur: Vincent Vliegen

Besluit

Auteurs: ; redactie: Arne Vlietinck

Referenties