



KU Leuven

Departement Computerwetenschappen

# P&O: COMPUTERWETENSCHAPPEN

## Tussentijds verslag

*Team:*  
**Zilver**

BRAM VANDENDRIESSCHE (COÖRDINATOR)

ARNE VLIETINCK (SECRETARIS)

MATTHIAS VAN DER HEYDEN

JEF VERSYCK

VINCENT VLIEN

LAURA VRANKEN

Academiejaar 2016-2017

## Inhoudsopgave

<b>1</b>	<b>Inleiding</b>	<b>2</b>
<b>2</b>	<b>Ontwerp</b>	<b>3</b>
2.1	Drone Autopilot . . . . .	3
2.2	Virtual Testbed . . . . .	5
<b>3</b>	<b>Algoritmes</b>	<b>5</b>
3.1	Drone Autopilot . . . . .	5
3.2	Virtual Testbed . . . . .	5
<b>4</b>	<b>Software</b>	<b>6</b>
4.1	Drone Autopilot . . . . .	6
4.2	Virtual Testbed . . . . .	6
<b>5</b>	<b>GUI</b>	<b>6</b>
5.1	Drone Autopilot . . . . .	6
5.2	Virtual Testbed . . . . .	7
<b>6</b>	<b>Testen</b>	<b>7</b>
6.1	Drone Autopilot . . . . .	7
6.2	Virtual Testbed . . . . .	8
<b>7</b>	<b>...</b>	<b>8</b>
<b>8</b>	<b>Besluit</b>	<b>8</b>
<b>A</b>	<b>Klassendiagram Autopilot</b>	<b>9</b>
<b>B</b>	<b>Klassendiagram Simulator</b>	<b>9</b>
<b>C</b>	<b>MOET NOG WEG</b>	<b>9</b>
<b>D</b>	<b>Beschrijving van het proces</b>	<b>9</b>
<b>E</b>	<b>Beschrijving van de werkverdeling</b>	<b>10</b>
<b>F</b>	<b>Kritische analyse</b>	<b>10</b>

# 1 Inleiding

*Auteur: 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, maar evenzeer voor ontspanningsdoeleinden. De bekendste toepassing bevindt zich binnen defensie die zo informatie kan winnen over vijandelijk gebied zonder mensenlevens te moeten riskeren. Daarnaast hebben ook grote bedrijven (e.g.: Amazon) 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, luchtfotografie etc. Om de verscheidenheid aan toepassingen te kunnen verwezenlijken, is natuurlijk een goede aansturing van belang. De drone kan aangestuurd worden via een persoon op de grond met een controller ofwel volledig autonoom. ([2])

Wanneer een drone autonoom functioneert, is een betrouwbare aansturing door de Autopilot van levensbelang. Hij moet namelijk bestand zijn tegen alle (eventueel extreme) weersomstandigheden. De twee basisdoelen van de Autopilot zijn: het doel bereiken op de meest efficiënte manier en de drone veilig laten landen op een aangegeven plaats.

Aangezien een autonome drone zijn informatie haalt uit de beelden die hij produceert, is de kwaliteit van de camera's van groot belang. Indien geen goed beeld verkregen wordt, is de relatieve positiebepaling moeilijker en minder precies te bepalen. Bovendien moet de Autopilot ook rekening houden met het verschil in intensiteit van de kleuren door bijvoorbeeld invloed van schaduw, felle zon en regendruppels op de camera. Door deze twee oorzaken kunnen toepassingen, waar nauwkeurigheid van belang is, hun nut verliezen.

In dit verslag gaan we verder in op hoe de autonome aansturing van een drone (en meer bepaald een quadcopter) gebeurt. We gaan uit van een drone waarop twee camera's vast bevestigd zijn op een zekere afstand van elkaar. Beiden zijn ze voorwaarts gericht. Op basis van deze beelden moeten afstand en positie van het doel ingeschat worden en nieuwe bewegingsopdrachten voor de drone gegenereerd worden. Aangezien er geen hardware ter beschikking was, moet er een Virtual Testbed simulator ontworpen worden. Dit is een softwaresysteem dat een fysieke opstelling van een drone en camera's simuleert. ([1]) 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 rode bol, te lokaliseren en er naar toe te vliegen. Dit eventueel onder lichte invloed van wind in willekeurige richtingen. Bovendien moet ook voor beiden een *GUI* ontworpen worden die ter beschikking staat van de gebruiker. De *GUI* toont de vooruitgang en laat de gebruiker toe allerlei informatie (snelheid, positie en verschillende camerastandpunten) op te vragen.

De tekst is als volgt opgebouwd. Eerst wordt het ontwerp van de Autopilot en Virtual Testbed verder uitgediept in sectie 2. Vervolgens wordt er ingegaan op de gebruikte algoritmen (sectie 3) en wordt de opbouw van onze software verduidelijkt (sectie 4). Ook wijden we uit over de toepassingen van de *GUI* in sectie 5. Tenslotte sluiten we in sectie 6 af met de uitgevoerde testen te bespreken, het besluit is te vinden in sectie 8.

## 2 Ontwerp

*Auteurs: ; redactie: Arne Vlietinck*

### 2.1 Drone Autopilot

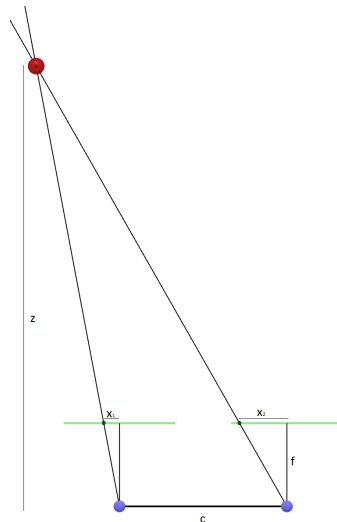
*Auteur: Laura Vranken*

De Drone Autopilot zorgt voor de besturing van de drone. Het bepaalt zijn positie relatief ten opzichte van zijn doel a.d.h.v. twee beelden. Deze beelden zijn gemaakt door twee camera's die op de drone gemonteerd staan op een vaste afstand van elkaar. Bovendien kunnen ze opgevraagd worden vanuit de GUI van het Virtual Testbed. Vervolgens zorgt de Autopilot ervoor dat de drone juist georiënteerd staat en naar zijn doel toe vliegt. Wanneer de drone zijn doel (in deze eerste fase is dit een rode bol) bereikt, moet hij daarin blijven zweven. Tenslotte moet de Autopilot ook rekening houden met een mogelijke invloed van wind die de drone van zijn koers doet afwijken.

Ten eerste moeten dus de beelden die de Autopilot van de Virtual Testbed binnen krijgt, geanalyseerd worden. Dit gebeurt door iteratief de integer waarden van elke pixel te vergelijken met de waarde van het kleur rood. Alle rode pixels worden bijgehouden door hun positie ten opzichte van het beeld, uitgedrukt in rij en kolom, op te slaan. We baseren onze komende berekeningen op het midden van de bol. Dat midden kan bepaald worden door het zwaartepunt van de rode pixels te berekenen via het gemiddelde van de opgeslagen coördinaten.

Indien de Autopilot geen rode pixels detecteert, zal de drone 360 graden ronddraaien of met andere woorden een yaw beweging uitvoeren, totdat in beide beelden rode pixels verschijnen. In het geval dat de Autopilot slechts rode pixels in een van de twee beelden opmerkt, zal het zichzelf door middel van een yaw beweging in de juiste richting bijsturen. Wanneer uiteindelijk in beide schermen rode pixels gevonden zijn, zal de Autopilot stoppen met draaien en zijn positie tegenover het doel berekenen.

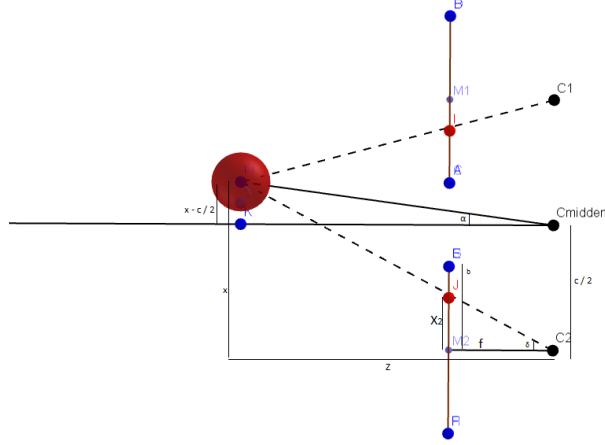
Om zijn positie tegenover het doel te berekenen, bepalen we eerste de diepte. Dit kan met behulp van de formule van stereo vision [3] uitgewerkt worden. Zie figuur 1 voor een grafische weergave van de berekening.



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

Vervolgens bepalen we de hoek waaronder de drone een yaw beweging moet uitvoeren om recht

naar het doel gericht te zijn. Deze formule kon afgeleid worden uit de goniometrie en wordt weer-gegeven in figuur 2.



Figuur 2: Relatieve horizontale hoek tussen drone en doel wordt gegeven door  $\tan(\alpha) = \frac{x - \frac{c}{2}}{z}$ , voor de volledige uitwerking zie formule 1 tot 4.

Om tenslotte naar het doel te kunnen vliegen, moet een evenwicht gevonden worden tussen pitch en thrust. De pitch hoek wordt gekozen zodat het zwaartepunt van het doel nog juist in beeld blijft. Die hoek is gelijk aan de helft van de verticale hoek die het beeld overspant min de verticale hoek waaronder de bol zich tegenover de drone bevindt, zie figuur 3.

Wanneer de pitch hoek vastligt, kan de hoeveelheid thrust berekend worden zodat de drone in rechte lijn naar het doel kan vliegen. Hieronder wordt de afleiding van de formule gegeven.

$$\frac{x_2}{f} = \frac{x}{z} \quad (1)$$

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

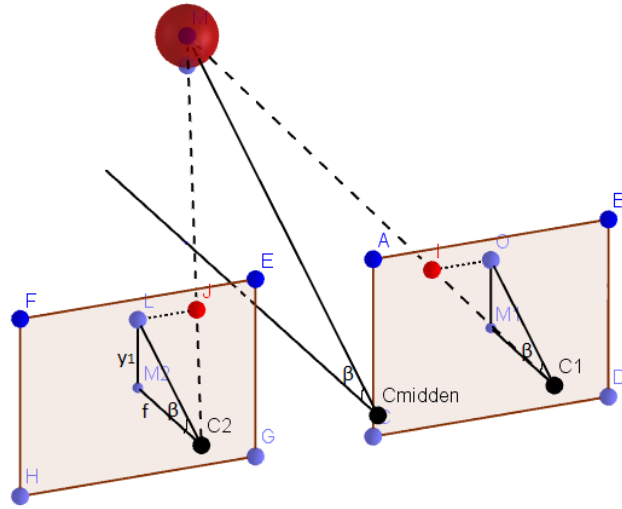
$$f = \frac{\frac{b}{2}}{\tan\left(\frac{\delta}{2}\right)} \quad (3)$$

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

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 moeten heroriënteren. Ook kan de wind ervoor zorgen dat de drone een roll uitvoert. Deze moet eerst gecompenseerd worden, vooraleer we verder onze berekeningen kunnen uitvoeren.

De drone bereikt zijn doel wanneer de Autopilot niks anders dan rode pixels opvangt. De drone zal dan de opdracht krijgen om zijn pitch te compenseren en vervolgens enkel via thrust de zwaartekracht tegen te werken.

Het effectief laten vliegen van de drone gebeurt in de Virtual Testbed waar de motoren worden aangestuurd. De Autopilot zendt enkel de verhouding in graden per seconden door waaronder pitch, yaw en roll moeten worden uitgevoerd en thrust in Newton. Het is slechts door herhaaldelijk te controleren hoe ver nog gedraaid moet worden, dat kan besloten worden wanneer de beweging volledig uitgevoerd is en wanneer er gestopt mag worden.



Figuur 3: Relatieve verticale hoek tussen drone en doel. De relatie wordt gegeven door volgende formule:  $\tan(\beta) = \frac{y_1}{f}$ .

Voor de volgende deadline is het de bedoeling om de beeldverwerking via *OpenCV* uit te voeren. Deze bibliotheek kan op een betere en gemakkelijkere manier vormen en kleuren herkennen. Dit speelt in ons voordeel wanneer er meerdere doelen met verschillende vormen en kleuren in beeld zijn. Hierdoor zal zowel het iteratief zoeken als het opslaan van de pixels vermeden worden. Aangezien *OpenCV* op een bepaalde range in RGB waarden kan zoeken, kan ook het probleem van lichtinval en schaduw opgelost worden. Ook wordt er gezocht naar een betere manier om de roll hoek in de berekeningen te trekken. Zo zal er tijd gewonnen worden aangezien die beweging niet altijd eerst moet gecompenseerd worden voordat de berekeningen kunnen worden uitgevoerd.

## 2.2 Virtual Testbed

*Auteur:*

## 3 Algoritmes

*Auteurs: ; redactie: Arne Vlietinck*

### 3.1 Drone Autopilot

*Auteurs:*

### 3.2 Virtual Testbed

*Auteurs:*

## 4 Software

*Auteurs: ; redactie: Arne Vlietinck*

### 4.1 Drone Autopilot

*Auteurs: Laura Vranken*

Tijdens het programmeren, is het belangrijk om een duidelijke structuur voor ogen te houden. Dit leidt immers tot goede leesbaarheid voor buitenstaanders en bewaart overzichtelijkheid bij lange code. Ook werd er geprogrammeerd met het doel om later gemakkelijk delen te kunnen hergebruiken. Om deze tekst beter te kunnen volgen, staat het klassendiagramma van dit deel in bijlage A.

Het deel van de autopilot begint bij het creëren van een autopilot in de *DroneAutopilotFactory* klasse, geïmplementeerd met de interface *AutopilotFactory*. Hierin wordt een autopilot aangemaakt en worden de beginwaarden voor yaw, roll, pitch en thrust direct ingesteld. De *GUI* wordt hier ook aangemaakt.

De autopilot wordt aangemaakt door middel van een nieuw object *DroneAutopilot*, geïmplementeerd met de interface *autopilot*, aan te roepen. De *DroneAutopilot* klasse bestaat uit één functie, namelijk *timeHasPassed* die continu door de simulator wordt uitgevoerd. Vanuit deze methode wordt de klasse *MoveToTarget* aangeroepen met de uit te voeren opdracht. In deze eerste milestone is het de bedoeling om naar de rode bol te vliegen.

De klasse *MoveToTarget* bepaalt de bewegingen en aansturing van de drone. Ook worden hier de verschillende rates doorgegeven aan de simulator. *MoveToTarget* steunt zowel op de informatie van de klasse *ImageCalculations* als van *PhysicsCalculations* om zijn bewegingen te bepalen. Bovendien wordt hierin ook telkens de *GUI* geüpdatet wanneer een nieuwe waarde voor de diepte bepaald is.

*ImageCalculations* is verantwoordelijk voor alles omtrent de beelden die de autopilot binnen krijgt, m.a.w. de rode pixels zoeken en het zwaartepunt bepalen. In de klasse *PhysicsCalculations*, worden tenslotte alle fysische berekening van hoeken en afstanden uitgevoerd.

### 4.2 Virtual Testbed

*Auteurs:*

## 5 GUI

*Auteurs: Arne Vlietinck; redactie: Arne Vlietinck*

Om de interactie en visualisatie gebruiksvriendelijk te maken, worden beide programma's voorzien van een *graphical user interface* (*GUI*). Er wordt gebruik gemaakt van Java's Abstract Window Toolkit (*java.awt*) om gemakkelijk de lay-out te programmeren. In de opgave van het project [1] staan er reeds richtlijnen waaraan beide *GUI*'s moeten voldoen.

### 5.1 Drone Autopilot

*Auteur: Matthias Van der Heyden*

De *GUI* van de drone autopilot heeft tot nu toe twee functies: de gebruiker de mogelijk geven een opdracht voor de drone te selecteren en de voortgang van de voltooiing van deze opdracht weergeven.

Voor het selecteren van een opdracht is er een dropdownmenu voorzien dat gebruik maakt van *JComboBox* uit de *Swing library*<sup>1</sup> van Java. Wanneer de gebruiker een optie aanduidt, verandert de boolean van deze optie naar true. Dit zorgt ervoor dat de juiste commando's voor deze opdracht uitgevoerd worden. Op dit moment is de enige optie in het menu de opdracht om naar de rode bol te vliegen maar een uitbreiding van mogelijke opdrachten voor volgende milestones is relatief eenvoudig.

Een *JProgressBar* uit de *Swing library* van Java geeft in de *GUI* de voltooiing van de geselecteerde opdracht weer. Bij elke berekening van de afstand tot het doel wordt deze geüpdatet. Is de afstand groter dan de laatste grootste afstand tot het doel dan stelt de progress bar deze nieuwe afstand in als het nieuwe maximum en is de voltooiingsgraad weer nul. Is de afstand kleiner dan is de nieuwe voltooiingsgraad gelijk aan 100 procent vermindert met de verhouding van de grootste afstand en de huidige afstand.

## 5.2 Virtual Testbed

*Auteurs: Arne Vlietinck*

Ook in het virtual testbed wordt een panel voorzien waarin de *GUI* vervat zit. In tegenstelling tot de autopilot wordt hier gebruik gemaakt van een iets uitgebreidere lay-outvorm namelijk de *GridBagLayout*<sup>2</sup>. Dit zorgt voor een op maat gemaakte lay-out, die noodzakelijk was voor het uitlijnen van de verschillende functies.

De centrale functie van deze *GUI* is het selecteren van verschillende camerastandpunten. Dit kan door middel van de *JButtons* die voorzien zijn. Naargelang de hoeveelheid camerastandpunten zullen er extra buttons gegenereerd worden.

Daarnaast kunnen ook de dronecamera's apart geselecteerd worden. Zo kan de gebruiker van het programma gemakkelijk de bewegingen van de drone vanuit verschillende standpunten beleven. Tot slot wordt ook de snelheid en de positie van de drone weergegeven. Deze wordt continu opnieuw opgevraagd en geüpdatet in de *GUI*.

## 6 Testen

*Auteurs: Arne Vlietinck; redactie: Arne Vlietinck*

Voor beide programma's worden er uitvoerig testklassen gegenereerd en uitgevoerd. Hiermee kan de correcte werking en de nauwkeurigheid van de gebruikte algoritmes gecontroleerd worden. Ook eventuele programmeerfouten komen aan het licht en kunnen op deze manier aangepast worden. Naar gelang de milestones uitdagender worden, zullen de testklassen striktere en hogere eisen stellen aan de geïmplementeerde tests.

### 6.1 Drone Autopilot

*Auteurs: Vincent Vliegen*

De autopilot is voorzien van enkele *JUnit* testklassen. Deze testen de verschillende gebruikte methodes op hun nauwkeurigheid en correctheid. Zo is het eenvoudiger de capaciteiten van de autopilot aan te tonen en de beperkingen duidelijk af te bakenen.

De *ImageCalculationsTest* is uitgerust met tests voor elke methode in de *ImageCalculations* klasse. Er is gebruik gemaakt van een anonieme klasse die de camera interface implementeert, zodat er afbeeldingen naar keuze gegenereerd kunnen worden. Dit vereenvoudigt immers het testen.

De *PhysicsCalculationsTest* verschaft testmethodes voor de *PhysicsCalculations* klasse. Ook hier zijn anonieme klassen geïmplementeerd, namelijk voor de camera en drone interfaces.

---

<sup>1</sup> *Swing library*: <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

<sup>2</sup> *GridBagLayout*: <https://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html>



## 6.2 Virtual Testbed

*Auteurs:*

## 7 ...

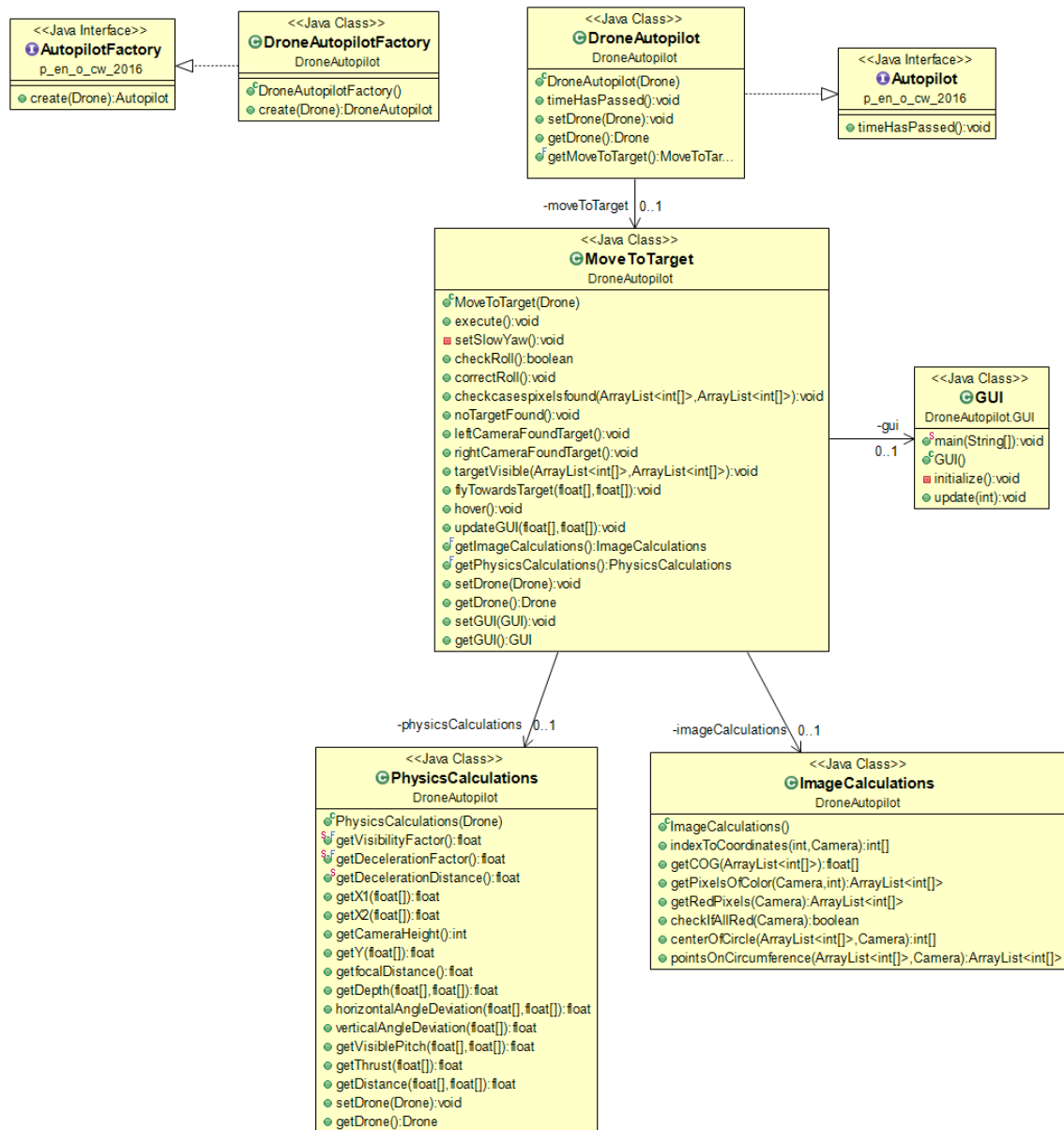
## 8 Besluit

*Auteurs: ; redactie: Arne Vlietinck*

## Referenties

- [1] H. BLOCKEEL, B. JACOBS, AND D. NUYENS, *Een automatische piloot en virtuele testomgeving voor drones*, 3 oktober 2016.
- [2] M. GMBH, *Microdrone-applications: aerial, photography, mapping, surveying, etc.* <https://www.microdrones.com/en/applications/>, 2016. [Geraadpleegd op 30 oktober 2016].
- [3] D. NAIR, *A guide to stereovision and 3d imaging*. <http://www.techbriefs.com/component/content/article/14925>, 1 oktober 2012. [Geraadpleegd op ???].

## A Klassendiagram Autopilot



## B Klassendiagram Simulator

## C MOET NOG WEG

De volgende informatie wordt na de finale demonstratie apart ingediend.

## D Beschrijving van het proces

- Welke moeilijkheden heb je ondervonden tijdens de uitwerking?
- Welke lessen heb je getrokken uit de manier waarop je het project hebt aangepakt?

