Documentatie

Info en uitleg van GLaDOS’ hart

Inhoud

[Inleiding 3](#_Toc424671183)

[Samenvatting 3](#_Toc424671184)

[Werking 4](#_Toc424671185)

[Herkenning 4](#_Toc424671186)

[Beweging 4](#_Toc424671187)

[Klassen 4](#_Toc424671188)

[Main 4](#_Toc424671189)

[Woordenlijst 5](#_Toc424671190)

# Inleiding

Deze documentatie is bedoelt voor personen die de werking van de code achter GLaDOS(Generiek LED afhankelijk drone opsporings systeem) willen begrijpen, of dit zelfs willen voortzetten.

Er zal worden ingegaan op wat iedere class doet en hoe ze met elkaar samenwerken om het tot een goed werkend geheel te maken.

Voor het leesgemak zullen methodes **vetgedrukt** zijn en variabelen *cursief*.

Ook zal er aan het einde een woordenlijst zijn, woorden die in de lijst voorkomen zijn onderstreept.

# Samenvatting

# Werking

## Herkenning

Bij de herkenning wordt in de setup de pi-cam gekoppeld aan *camera*.

Daarna wordt **Java\_DroneTracker\_Track** aangeroepen. Vanuit hier word de werkelijke herkenning aangestuurd en deze zal iedere keen aangeroepen moeten worden om te herkennen wat er in beeld is van de pi-cam. De boolean *tracker* geeft aan of er een object is herkend. Dan zal het camerabeeld omgezet worden naar HSV.

Dan is er een for-loop die 2 keer doorlopen zal worden, dit is om op de 2 verschillende kleuren led’s van de drone te zoeken. De eerste keer wordt er gezocht naar de blauwe led’s, de 2e naar de rode led’s.

In de while-loop is een zoek algoritme verwerkt. De methode **trackFilteredObject** kan 3 waardes terug geven:

1. Een int met waarde 0, dan is de drone gevonden.
2. Een int met waarde 1, dan is er teveel ruis in beeld en zal dit met morphOps worden weggehaald. morphOps haalt de ruis weg, en zal wat over blijft vergroten. Dit zal maximaal 3 keer gebeuren, daarna zal het niet veel nut meer hebben en dit voorkomt dat het hier in een oneindige lus terecht komt.
3. Een int met waarde 2, dan is er geen object gevonden en waren er ook te weinig objecten met de juiste kleur om überhaupt op zoek te gaan naar een object.

In **trackFilteredObject** zal er naar een object worden gezocht als er in de *threshold* ten minste 1 object is. Daarna zal er van ieder object een x en y in een array gestopt worden voor later gebruik.

## Beweging

Het volgen van de drone zal gebeuren als **trackFilteredObject** iets vind zal de *lastKnownTime* geüpdate worden naar de huidige tijd. Daarna zal er naar de ServoController **Update** een Point gestuurd worden.

Als er niets word gevonden zal er eerst worden gekeken hoelang het geleden is dat de drone werd gevonden, als dit meer is dan 2,5 seconden geleden is zal GLaDOS overgaan op idle-modus en rond gaan bewegen totdat er weer een drone word gevonden. Iedere 150 iteraties zal er een volgend punt naar de ServoController **Update** gestuurd worden, deze vertraging is er zodat de arm eerst naar het opgegeven punt kan bewegen, voordat het naar de volgende gaat.

In de klasse ServoController staat een lijst met pinnen, deze verwijzen naar de 4 servo’s die in de arm verwerkt zijn. Iedere pin heeft een fysiek pin nummer, waarmee pi-blaster weet naar welke pin het PWM moet worden geschreven, ook staan er de startpositie, informatie over de servo.

In de methode **Update** wordt gekeken of het gekregen punt buiten de deadzone ligt, dit is om te voorkomen dat de arm bij iedere kleine beweging mee gaat met de drone, en hierdoor erg stotterig word. Als het buiten de deadzone ligt wordt respectievelijk de x of de y naar PWMPin verstuurd worden. In PWMPin.**Update** word berekend wat er naar pi-blaster gestuurd moet worden om de drone in beeld te houden. Dit word weggeschreven naar het pi-blaster bestand op de pi.

# Klassen

## Main

# Woordenlijst

HSV Variabele van OpenCV, deze zorgt voor grotere kleurverschillen tussen kleuren die in BGR dicht bij elkaar zitten