Zoeken, Sturen & Bewegen

Technisch Rapport

Follow The Target:
Kleurherkenning met de Parrot AR-Drone 2.0

Auteurs:
Tim Ottens (11147598)
Max Teeuwen (10834516)
Julius Meetsma (11295368)
Mattijs Blankesteijn (11318422)

Tutor:
Ghislaine Van den Boogerd

Universiteit van Amsterdam

Juni 2017

Abstract

Dit onderzoek beschrijft hoe een vliegend object acties kan uitvoeren afhankelijk van de kleur van ondergrond. De AR drone 2.0 van Parrot wordt in dit experiment gebruikt. Er is voor deze drone gekozen omdat deze kan vliegen en over een camera beschikt die beelden *live* kan *streamen*. Uniek aan dit onderzoek ten opzichte van eerder onderzoek is dat er hier een lijn wordt gevolgd op verschillende hoogtes door gebruik te maken van kleurherkenning. De verschillende kleuren zijn een representatie voor obstakels die in de weg staan, waardoor de drone de weg niet verder kan volgen in een echte situatie. Deze techniek kan onder andere toegepast worden in het transport, het verkeer en andere lijn- en/of kleurgebaseerde maatschappelijke elementen.

Inleiding

Drones worden steeds meer ingezet om verschillende taken van mensen te vervullen. Een voorbeeld is het gebruik van drones om pillen en verbanddozen in ziekenhuizen te vervoeren zodat hiermee tijd kan worden gewonnen voor belangrijkere zaken waarbij wel expertise nodig is. Deze toepassingen gebeuren ook voornamelijk vanwege efficiëntie en kostenbesparingen. De volgende stap in dit proces zijn autonome drones. Hiervoor is het niet nodig dat een mens deze drones bestuurt. De transportsector is een van de gebieden waarin autonome drones kunnen bijdragen aan een efficiënter en goedkoper productieproces. Hier is veel potentie voor het toepassen van drones.

Een concept dat handig kan zijn in verschillende gebieden van transport is een drone die volgens een gekleurd pad een object kan leiden naar zijn bestemming. Benodigdheden voor een dergelijke applicatie zijn onder andere kleurherkenning, automatisering van beweging en objectherkenning. Verschillende projecten zijn ons voorgegaan in padvolging, maar geen van de projecten heeft dit gekoppeld aan beweging op basis van kleurherkenning. Het is de bedoeling dat de drone op basis van de verschillende kleuren verschillende acties kan uitvoeren. Het op verschillende hoogtes vliegen van de drone kan gebruikt worden om objecten te vermijden die vast op de grond staan of om door ingewikkelde ruimtes te navigeren. De hoogte van de drone kan bepaald worden op basis van de kleurverschuiving van de ene kleur naar de ander.

Dit alles in overweging nemende kwamen wij tot de volgende onderzoeksvraag:

> *In hoeverre is de* Parrot AR-drone 2.0 *capabel om aan de hand van kleurherkenning verschillende acties uit te voeren binnen een gegeven ruimte?*

De bijbehorende hypothese houdt in dat de drone tot dusver capabel is dat hij in een eenvoudige omgeving meerdere soorten kleuren zou moeten kunnen herkennen. Aan deze kleuren moet de drone ook de juiste acties koppelen, zowel laagvliegend als in hogere omgevingen. Naast een hypothese roept deze onderzoeksvraag ook deelvragen op als

1. *Op welke wijzen is de Parrot AR-drone 2.0 dusdanig vanaf een computer aan te sturen dat er naar tevredenheid mee te werken is?*
2. *Wat voor software is er nodig voor kleurherkenning en op welke manier moet dergelijke software geïmplementeerd worden?*
3. *Hoe moet de drone geprogrammeerd worden zodat deze op basis van kleur- en patroonherkenning een lijn kan volgen en bijbehorende acties kan uitvoeren?*

Verwachtingen bij deze deelvragen zijn onder andere dat met behulp van de juiste software (TheNodecopterteam [5] & OpenCV-team [3]) de drone op een acceptabele manier te controleren is, zowel in bewegingen als in analyse van binnenkomende beelden. De uiteindelijke verwachting is dat er door het implementeren van de bovengenoemde software en het gebruik van de juiste hardware een eindresultaat bereikt kan worden die herkenbare elementen bezit naar het idee van een lijn-volgende drone.

Er bestaan veel experimenten met allerlei soorten drones die een lijn volgen [2][6]. Dit is echter niet eerder gecombineerd met acties op basis van kleurherkenning zoals in dit onderzoek. Ook wordt er in dit experiment rekening gehouden met verschillende kleuren lijnen, die representatief zijn voor een verschil in grond, of voor obstakels op de weg waar de drone overheen beweegt.

Ons onderzoek is een representatie voor een werkelijke situatie waarin er objecten vervoerd moeten worden met obstakels in de weg. Het is vaak efficiënter om een object over de grond te transporteren dan door de lucht. Echter, volledig transport over de grond is niet altijd mogelijk, door obstakels die in de weg kunnen staan. In een dergelijke situatie kan het pragmatisch zijn om op te stijgen en over het obstakel heen te vliegen.

Dit zou in de nabije toekomst kunnen worden toegepast in de gehele transportsector, zoals bij bezorgingsdiensten. Ook zou de essentie van dit onderzoek gebruikt kunnen worden in het verkeer. Denk aan een applicatie in een auto die de weg waarneemt, en hierop de snelheid of de stuurcorrectie aanpast. Dit kan bijdragen aan de ontwikkeling van een automatische piloot, maar ook voor het efficiënter benutten van brandstof.

Methode

Voor dit project gebruiken wij de AR-Drone 2.0 met ingebouwde camera *feed*. De eerste stap is om de beweging van de drone te automatiseren. Dit wordt gedaan met behulp van een programma gemaakt in Javascript dat Nodecopter [5] heet. Dit vanwege het feit dat het direct vanuit de handleiding [7] werd aangeraden. De commands van de drone worden in Javascript geprogrammeerd. Ook wordt hiermee toegang aangevraagd tot de camera van de drone.
De camera van de drone is nodig voor kleurherkenning die wordt geïmplementeerd met OpenCV in de programmeertaal Python. [3] In OpenCV krijgen we de camera feed door van de drone en verdelen het inkomende beeld in 25 (5x5) vakjes. Hier hebben wij een oneven getal gekozen om zo een midden te creeëren, 7x7 was ook een prima keuze geweest maar dit zou onnodig veel meer computatievermogen eisen. OpenCV heeft kleurlimieten opgekregen die alleen kleuren binnen deze limieten laat zien via de camera. De lower en upper **boundaries** voor de verschillende kleuren in het BGR-kleurenschema zijn:

Figuur 1: De drone tijdens het experiment

Figuur 2: De opstelling van het experiment

- *rood* ([15,15,160], [120,120,255])
- *zwart* ([1,1,1],[65,60,65])
- *wit* ([190,190,190],[255,255,255])

Deze kleuren zijn gekozen omdat ze in onze opstelling relatief de beste resultaten opleverden. Vervolgens wordt op basis van de aanwezigheid van dit kleurschema in elk vakje een keuze gemaakt welke kant de drone op moet gaan. De voorkeur van de drone is ingesteld op een voorwaartse beweging zodat er bij zowel input in de linkerrij van het scherm als middelste rij de drone eerst vooruit vliegt. Als in deze richting geen kleur wordt gedetecteerd kijkt de drone vervolgens naar de hele eerste rij van boven in zijn scherm en als deze geen kleur bevat kijkt hij naar de volgende rij enzovoorts. De richting wordt doorgegeven aan de Node.js die de beweging ontvangt van het Python bestand. Om informatie door te geven van Python naar Javascript wordt er een package gebruikt namelijk Python-Shell.

Wanneer er geen kleur wordt ontdekt van het huidige kleurenschema in de voorwaartse beweging wordt er van kleurenschema gewisseld in OpenCV. Als er wel wat wordt gedetecteerd in dit kleurenschema betekent dit dat de drone lager of hoger moet vliegen op basis van deze kleur. Dit wordt weer doorgegeven van Python naar het Javascript-command bestand.
Deze acties blijft de drone herhalen totdat het doel is bereikt. Het doel kan ook weer worden aangegeven met een aparte kleur waarbij de drone zijn landing inzet.

De Code

De volgende (onderstaande) code geeft weer hoe de drone werd aangestuurd vanuit de laptop (zeer vereenvoudigd). In de praktijk is dit een uitgebreid NodeJS-script waar voor elke soort kleur en beeldlocaties die het algoritme binnenkrijgt een statement is die daar de juiste beweging aan koppelt. Bijvoorbeeld: *als er in de middelste drie vakken wit te zien is mag ik vooruit* en *als ik ergens rood zie moet ik landen.*

```
// Controlling the drone (pseudocode from the nodejs)

// Gebruik downcamera
Drone.(videokanaal, 3)

// Start
if key == s then
    Drone.takeoff
    newmove()

if key == q then
    Drone.land

// As long as new move is heard
function newmove:
    script = scriptpath.connect
    script.on('message')
        if 'message' contains right values:
            execute move for that message
```

Zoals uit bovenstaande code volgt word er een pythonscript aangeroepen dat de actuele meest belangijke kleur (landen is belangrijker dan doorvliegen) samen met zijn posities in het zichtveld meegeeft. Vereenvoudigt komt dat op het volgende neer:

```
// Understanding the images of the drone (pseudocode from the pythonscript)

cam = cv.VideoCapture('DroneStreamIP')
colors = [(lower, upper)]

while(true):
    frame = cam.read
    // For every color we want to check perform this loop
    for(color in colors):
        //We have to remove all color that is outside the RGB values
        mask = maskColor(frame);
        // Then we devide the screen into a set of panels that count how
        // many colored pixels are present
        squaredScreen(mask);
        //from that information we can decide what movement we want to
        // return to the Javascript
        nextLocationToMove();

        if(noAreaColored):
            Nextcolor;

    // Begin with most important color for next frame again
    color = firstcolor;
```

Resultaten

Het uiteindelijke doel van dit project was dat de drone verschillende kleuren lijnen kon herkennen en op basis van deze kleuren acties uit kon voeren. Tevens zou de drone een object moeten kunnen oppakken en deze met zich mee brengen. Echter, is er in dit project alleen aandacht besteedt aan het lijnvolgen en acties uitvoeren op basis van de verschillende kleur lijnen. Objectherkenning en transport zouden eventueel in vervolgonderzoek moeten worden bekeken.

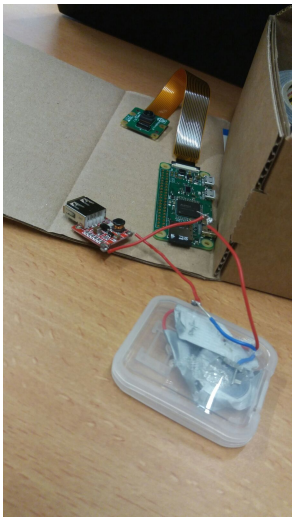
Het behaalde resultaat is dat de drone zoals beschreven verschillende kleuren lijnen kan detecteren en volgen, en hierbij ook acties uitvoert. Het opstijgen van de drone was echter niet altijd stabiel waardoor een poging tot lijnvolgen meteen ongedaan werd gemaakt. Een afwisselende oplossing voor dit probleem was een kalibratie die we drone lieten uitvoeren waardoor deze zijn balans weer kon vinden. De andere oplossing was het vervangen van de propellers die na een paar gefaalde pogingen verbogen waren.

De origineel gebruikte kleuren voor de lijnen waren wit en blauw. Na veel pogingen om het blauw goed te laten detecteren door de camera van de drone bleek de kleur geen goede keuze en is er een zwarte lijn gekozen. Deze kleur lijn leverde een beter resultaat op aangezien deze eerder werd herkend en een vollere lijn weer gaf op de camera. De landing werd ingezet als de camera een rood object detecteerde. Het rode object had prioriteit over zowel de zwarte als witte lijn en de zwarte lijn had prioriteit over de witte lijn. Alle rode objecten werden snel gedetecteerd door de camera en leverde in vergelijking met de andere kleuren een goed resultaat op in zowel tijd als volledige detectie.

Uiteindelijk bleek het moeilijk om de bewegingen van de drone zo in te stellen dat deze zich zou stabiliseren als deze schuin ten opzichte van de lijn stond of naast de lijn. Om dit gedaan te krijgen moest er een perfecte combinatie worden gevonden tussen het aantal opdrachten per tijdsinterval en de snelheid van de bewegingen.

Discussie

Veel van de oorspronkelijke ideeën uit het onderzoeksvoorstel konden niet in de praktijk worden uitgevoerd. In het onderzoeksvoorstel wordt gesproken over de RollingSpider-drone die een lijn moet volgen. Dit zou moeten gebeuren door een camera onder in de drone, die de beelden verstuurd naar een programma die checkt of er een bepaalde kleur onder zit, en aan de hand hiervan bepaald wat voor move de drone moet uitvoeren. Echter, in de werkelijkheid is het (nagenoeg) niet mogelijk voor de camera onder in de drone om de beelden direct te versturen. Naast een vertraging (*Bluetooth Low Energy*), is ook de softwarmatige ondersteuning dermate onbruikbaar dat die het onmogelijk maakt voor de drone om een pad te volgen. Dit probleem is verholpen door een andere drone te gebruiken, de AR drone 2.0. Deze drone voldoet aan de eisen die nodig zijn voor het uitvoeren van dit onderzoek, alhoewel die wel minder stabiel kon vliegen in de relatief kleine ruimte van het robolab.



Figuur 3: De opstelling van de Raspberry Pi

Om het eerder genoemde probleem met de ca-

mera van de Parrot RollingSpider op te lossen, is er overwogen om een externe camera toe te voegen. Deze camera zou een module van de Raspberry Pi Zero W gebruiken. Dit is een camera van 3 gram. Voor de RollingSpider is dat gunstig, want deze drone is licht en beschikt niet over veel draagkracht. Het probleem dat werd gecreëerd met deze methode, is dat de camera niet geheel draadloos is. Om het te laten werken is er een powerbank vereist. Het nadeel met een dergelijke powerbank is het gewicht. Om dit te voorkomen is er voor dit onderzoek een batterij ontworpen. Dit zijn twee 3V knoop batterijen in serie geschakeld, die worden getransformeerd naar 5V. Dit zou in theorie de energie moeten leveren die nodig is voor de Raspberry Pi. Echter, in de praktijk is deze stroomtoevoer helaas niet stabiel genoeg; na het opstarten van de Raspberry Pi valt deze na enkele seconden weer uit door een tekort aan energie.

Als de drone te veel schommelt, kan deze moeite hebben met het identificeren van een patroon. Daarom is het van belang dat de drone stabiel in de lucht hangt om een lijn succesvol te volgen. Dit probleem is aangepakt door van de RollingSpider over te stappen naar de AR drone 2.0. Deze drone is beter in staat om recht te blijven vliegen dan de RollingSpider, echter blijft de AR drone 2.0 ook niet geheel stabiel in de lucht.

Dat de drone soms nog moeite heeft met het volgen van de lijn ligt met name aan het feit dat het een vrij complexe taak is om zelf aan te geven voor welke kleuren en welke gebieden een drone een bepaalde beweging moet doen. Elke keer als er een beweging werd toegevoegd kwam werd er geconstateerd dat er nog meer bewegingen nodig waren. Bij toekomstig onderzoek lijkt het ons interessant om met *(supervised) machine learning* de drone zelf te (laten) leren welke bewegingen er bij welk beeld gemaakt moeten worden om een lijn recht te volgen. Andere verbeterpunten zijn onder andere het eerder wisselen van bepaalde hardware als duidelijk is dat er niet mee te werken is. In dit onderzoek is relatief veel tijd verloren aan het proberen werkend te krijgen van de Rolling Spider. Tenslotte dagen wij toekomstige onderzoekers uit om een stabielere versie van de software maar ook zeker van de gebruikte hardware te ontwikkelen.

Referenties

[1] T. Carpay, T. Cools, M. Fennema, K. Molenaar, and T. Wagenaar. Touwbruggen gemaakt door een drone. *University of Amsterdam*, 2017. URL `https://staff.fnwi.uva.nl/a.visser/education/ZSB/2016/RollingSpider_Labboek_Week_4.pdf`.

[2] A. Hernandez, C. Copot, R. De Keyser, T. Vlas, and I. Nascu. Identification and path following control of an ar. drone quadrotor. In *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*, pages 583–588. IEEE, 2013.

[3] OpenCV-team. Opencv. 2017. URL `http://opencv.org/`.

[4] Parrot. Parrot sdk3. 2017. URL `http://developer.parrot.com/docs/SDK3/`.

[5] The-Nodecopterteam. The nodecopter. 2012. URL `http://www.nodecopter.com/`.

[6] C. R. Verschoor and A. Visser. Integrating disparity and edge detection algorithms to autonomously follow linear-shaped structures at low altitude. In *AI & Robotics and 5th RoboCup Iran Open International Symposium (RIOS), 2013 3rd Joint Conference of*, pages 1–7. IEEE, 2013.

[7] A. Visser. Course website. *University of Amsterdam*, 2017. URL `https://staff.fnwi.uva.nl/a.visser/education/ZSB`.

Appendix

Omdat we het toch jammer vinden om het werk dat verzet is om de Parrot Rolling Spider aan de praat te krijgen zomaar weg te gooien geven we hierbij nog de code om deze drone te besturen (met behulp van een extern OpenCV-script dat nagenoeg hetzelfde werkt als het uiteindelijke pythonscript.

```
/*****************************************************************************
 *                                                                           *
 * file: followthetarget.js                                                  *
 *                                                                           *
 * Zoeken, Sturen & Bewegen 2017                                             *
 * Universiteit van Amsterdam                                                *
 *                                                                           *
 * Project Follow The Target                                                 *
 * Mattijs Blankesteijn, Max Teeuwen, Julius Meetsma, Tim Ottens             *
 *                                                                           *
 * This file contains a javascript implementation for controlling the Parrot *
 * Rolling Spider. The drone is indirectely controlled by an OpenCV script   *
 * made in Python. This file is based on the keyboard.js file from the       *
 * the node-rolling-spider libary (github)                                   *
 *                                                                           *
 *****************************************************************************/


'use strict';

var keypress = require('keypress');
var Drone = require('rolling-spider');
var scriptpath = 'opencv.py';

var ACTIVE = false;
var STEPS = 2;


function cooldown() {
  ACTIVE = false;
  setTimeout(function () {
    ACTIVE = true;
  }, STEPS * 12);
}

console.log(''
+  '*****************************************************************************'
+'\n*                                                                           *'
+'\n*  Welcome to the automatic drone controlling script of Follow The Target by:  *'
+'\n*      Tim Ottens (11147598)                                                *'
+'\n*      Max Teeuwen (10834516)                                               *'
+'\n*      Julius Meetsma (11295368)                                            *'
+'\n*      Mattijs Blankesteijn (11318422)                                      *'
+'\n*                                                                           *'
+'\n*  Press "s" to start, press "ctrl + c" to exit                             *'
+'\n*                                                                           *'
+'\n*  University of Amsterdam | Zoeken, Sturen & Bewegen | Juni 2017           *'
+'\n*                                                                           *'
+'\n*****************************************************************************'
+'\n')

// Make `process.stdin` begin emitting 'keypress' events
keypress(process.stdin);

// Listen for the 'keypress' event
process.stdin.setRawMode(true);
process.stdin.resume();

if (process.env.UUID) {
  console.log('Searching for ', process.env.UUID);
}

// Connect and setup drone
var d = new Drone();
d.connect(function () {
  d.setup(function () {
    console.log('Configured for Rolling Spider! ', d.name);
    d.flatTrim();
    d.startPing();
    d.flatTrim();
```

```
71      console.log(d.status.battery + '% battery left' )
72
73      setTimeout(function () {
74        console.log('ready for flight, press "s" to takeoff!');
75        ACTIVE = true;
76      }, 1000);
77    });
78  });
79
80  function execute(move) {
81      if (move === 'm') {
82          d.emergency();
83          console.log('Emergency! Shutting down drone and disconnect ...')
84          setTimeout(function() {
85              process.exit()
86          }, 3000)
87      }
88      else if (move === 'w') {
89          d.forward({ steps: STEPS });
90          cooldown();
91      } else if (move === 's') {
92          d.backward({ steps: STEPS });
93          cooldown();
94      } else if (move === 'left') {
95          d.turnLeft({ steps: STEPS });
96          cooldown();
97      } else if (move === 'a') {
98          d.tiltLeft({ steps: STEPS });
99          cooldown();
100     } else if (move === 'd') {
101         d.tiltRight({ steps: STEPS });
102         cooldown();
103     } else if (move === 'right') {
104         d.turnRight({ steps: STEPS });
105         cooldown();
106     } else if (move === 'up') {
107         d.up({ steps: STEPS * 2.5});
108         cooldown();
109     } else if (move === 'down') {
110         d.down({ steps: STEPS * 2.5 });
111         cooldown();
112     } else if (move === 'i' || move === 'f') {
113         d.frontFlip({ steps: STEPS });
114         cooldown();
115     } else if (move === 'j') {
116         d.leftFlip({ steps: STEPS });
117         cooldown();
118     } else if (move === 'l') {
119         d.rightFlip({ steps: STEPS });
120         cooldown();
121     } else if (move === 'k') {
122         d.backFlip({ steps: STEPS });
123         cooldown();
124     } else if (move === 's') {
125         d.backward({ steps: STEPS });
126         cooldown();
127     } else if (move === 'q') {
128         console.log('Initiated Landing Sequence...');
129         d.land();
130     }
131 }
132
133 process.stdin.on('keypress', function (ch, key) {
134     if (!ACTIVE && key && key.name === 's') {
135         console.log('Waiting for connection with drone ...')
136     }
137     else if (ACTIVE && key && key.name === 's') {
138         ACTIVE = false;
139
140         function newmove() {
141             // Use python shell
142             var PythonShell = require('python-shell');
143             var pyshell = new PythonShell(scriptpath);
144
145             pyshell.on('message', function (message) {
146                 // received a move sent from the Python script
147                 execute(message)
148             });
149
150             // end the input stream and allow the process to exit
```

```
151             pyshell.end(function (err) {
152                 if (err){
153                     throw err;
154                 };
155             });
156         }
157 
158         // Actual start
159                 console.log('takeoff');
160                 d.takeOff();
161 
162         // Listen for new moves
163         newmove()
164 
165         // Exit after last move
166         console.log('Press Ctrl + c when landed to exit')
167     }
168 
169     // Exit the program
170     if (key && key.ctrl && key.name === 'c') {
171         console.log('Bye!');
172         process.stdin.pause();
173         process.exit();
174     }
175 
176     // Emergency
177     if (ACTIVE && key && key.name === 'm') {
178         console.log('Trying emergency landing ... ');
179         execute('q');
180         setTimeout(function() {
181             execute('m');
182         }, 2000)
183         setTimeout(function() {
184             process.stdin.pause();
185             process.exit();
186         }, 2000)
187 
188     }
189 });
```