Westfälische Hochschule

University of Applied Sciences
Gelsenkirchen Bocholt Recklinghausen

Projektarbeit

Titel der Arbeit // Title of Project

Regelung der Flugposition eines Nanocopters mittels Computer Vision

Controlling the hover position of a nanocopter via computer vision

---

Autorenname, Matrikelnummer // Name, Matriculation Number

Sebastian ######, ##########

Studiengang // Course of Study

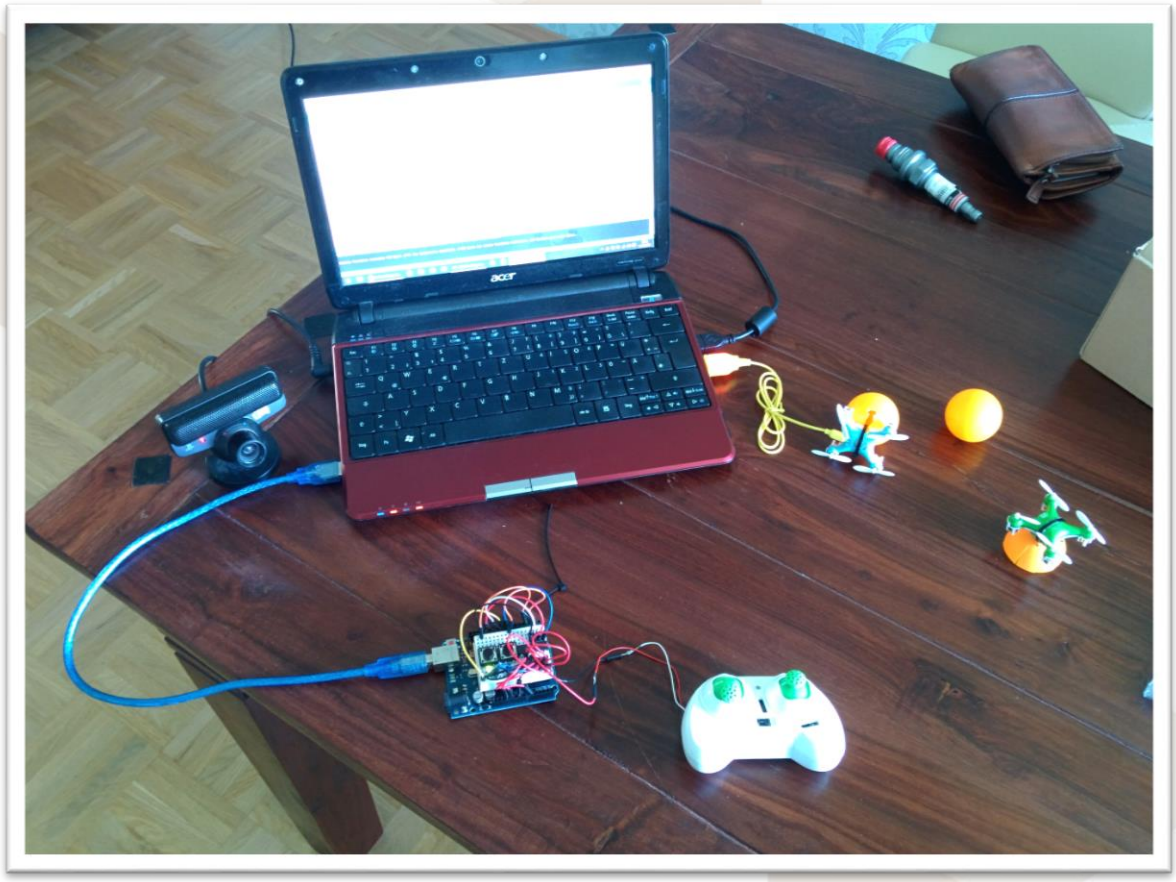
Mechatronik (B. Eng.)

Fachbereich // Department

FB6 Maschinenbau

Prämisse des Projektes

Ziel dieses Projektes war die Regelung der Flugposition eines Nano-Quadcopters (Cheerson CX-10) durch das Erkennen des Quadcopters im Kamerabild einer handelsüblichen USB-Kamera (PlayStation Eye Kamera). Zunächst war nur die Regelung in einer Raumachse bezogen auf die Flughöhe des Fluggerätes geplant. Dies sollte Anfängern das Fliegen erleichtern, indem diese nur links-rechts sowie vor-zurück, nicht aber auf-ab steuern müssen. Letztendlich habe ich mich jedoch für die Regelung von links-rechts und vor-zurück entschieden, da hier beide Koordinatenachsen des Kamerabildes zum Tragen kommen. Der X-Wert der Kamerabildpunkte korrespondiert mit der Flugsteuerung links-rechts und der Y-Wert bezieht sich auf vor-zurück. Die Kamera wird auf dem Boden platziert und zeigt in Richtung Decke. Besonders interessiert hat mich der Einsatz eines Arduino-Boards als Hardware-Schnittstelle sowie die Kommunikation zwischen einem PC und anderer Peripherie.



Versuchsaufbau des Projektes

Komponenten des Projektes

- Hardware
  - Cheerson CX-10 Nano-Quadcopter
  - PlayStation Eye Kamera
  - Arduino Uno R3
  - Digital-Potentiometer MCP4261
  - Orangefarbende Tischtennisbälle
- Software
  - Selbstgeschriebenes OpenCV-Programm (C++)
  - Selbstgeschriebenes Arduino-Programm (C++)

Cheerson CX-10 Nano-Quadcopter

Dieser Nanocopter war bis vor kurzem der kleinste Quadcopter der Welt. Er wird mit einer 2,4GHz-Funkfernbedienung gesteuert und liegt mit Hilfe von integrierten Gyroskopsensoren stabil in der Luft. Sechs Achsen lassen sich steuern und der 3,7V 100mAh Li-Ionen–Akku sorgt für eine Flugzeit von etwa 5min ohne Zusatzlast. In der Konfiguration für das Projekt mit befestigter Tischtennisballhälfte, bedingt durch das Zusatzgewicht, nur etwa halb so lange. Die Abmessungen sind 4,2 x 4,2 x 2 cm (L x B x H) und das Gewicht ohne Last beträgt 29g.

Playstation Eye Kamera

Die PlayStation Eye Kamera ist in der Lage Videos mit einer Auflösung von 640x480 mit 60 Bildern pro Sekunde aufzunehmen. Sie ist als Zubehör für die gleichnamige Spielekonsole erhältlich und eignet sich durch die hohe Framerate besonders für schnelle Gesteneingaben bei Spielen. Mittels spezieller Treiber ist diese Kamera auch unter Windows am PC zu betreiben und wird dank der genannten Spezifikationen gerne für Computer-Vision-Applikationen eingesetzt. Der Anschluss erfolgt über USB.

Arduino Uno R3

Das Arduino Uno ist ein Mikrocontroller-Board basierend auf dem ATmega328. Er hat 14 digitale Ein-/Ausgabe-Pins (wovon 6 als PWM-Ausgänge verwendet werden können), 6 analoge Eingänge, einen 16 MHz Keramikresonator, einen USB-Anschluss, eine Power-Buchse,

einen ICSP-Header und eine Reset-Taste. Der Uno kann direkt über den USB Anschluss programmiert werden und stellt in diesem Projekt die Brücke zwischen dem Computer mit der Computer Vision Software und der Fernbedienung des Nanocopters her. Er empfängt hierfür die X- und Y-Koordinate des getrackten Objektes über den Serial Port und regelt durch einen PID-Regler den Ausgang eines Digital-Potentiometers, welches wiederum an ein Analog-Potentiometer in der Fernbedienung angeschlossen ist. Eine genauere Beschreibung dieses Prozesses findet sich im Code des Arduino-Programms.

Digital-Potentiometer MCP4261

Das MCP4261-Digital-Potentionmeter hat zwei Kanäle, ist SPI-gesteuert und die Steuerung funktioniert mit zwei Bytes. Das erste Byte sendet die Adresse des Poti-Kanals (Poti 0 oder Poti 1) und das zweite Byte sendet den Widerstandswert, der von 0 bis 255 eingestellt werden kann. Es hat einen Einstellbereich von 0 bis 10kOhm und wird parallel zu dem Potentiometer in der Fernbedienung geschaltet.

Orangefarbene Tischtennisbälle

Die hellorangefarbenen Tischtennisbälle fungieren als Marker für den Nanocopter. Das Tracking des Objektes im Kamerabild funktioniert durch die Filterung einer bestimmten Farbe und die genannte Farbe eignet sich besonders gut, da sie leicht vor verschiedenen Hintergründen erkennbar und somit gut filterbar ist.

OpenCV-Programm

Das OpenCV-Programm wurde von mir im Rahmen des Programmierprojektes für das Fach Technische Informatik (TINF) entwickelt und für das Nanocopter-Projekt angepasst und mit der Arduino-Kommunikation erweitert. Die Präsentation mit der genauen Funktionsbeschreibung findet sich im Anhang (oder unter folgendem Link: http://bit.ly/1FG0NJ7).

Die Kommunikation mit dem Arduino-Board erfolgt über die serielle Schnittstelle und ist folgendermaßen implementiert:

```
void arduinoSerial(int x, int y)
{
	char x_wert[4];
	sprintf(x_wert,"%d",x);
	x_wert[3] = 42;

	char y_wert[4];
	sprintf(y_wert,"%d",y);
	y_wert[3] = 43;

    // Deklariert Variablen und Strukturen
    HANDLE hSerial;
    DCB dcbSerialParams = {0};
    COMMTIMEOUTS timeouts = {0};

    // Einstellen des COM-Ports
    hSerial = CreateFileA("\\\\.\\COM4", GENERIC_READ|GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL );

    // 38400 baud, 1 Start bit, 1 Stop bit, No Parity
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    dcbSerialParams.BaudRate = CBR_38400;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    SetCommState(hSerial, &dcbSerialParams);

    // Setzt COM-Port timeout-Einstellungen
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;
    SetCommTimeouts(hSerial, &timeouts);

    // Sendet die X- und Y-Koordinate an das Arduino Board
      WriteFile(hSerial, x_wert, 4, NULL, NULL);
      WriteFile(hSerial, y_wert, 4, NULL, NULL);

    CloseHandle(hSerial);
}
```

Das char-Array für den X-Wert hat drei Bytes für den dreistelligen Koordinatenwert und endet mit dem Byte-Dezimalwert 42, welches ein [*] darstellt. Das Array des Y-Wertes arbeitet genauso und endet mit dem Byte-Dezimalwert 43, welches ein [+] darstellt. Mit der Endung kann auf der Empfängerseite im Arduino der Zahlenwert der richtigen Koordinate zugeordnet werden.

Arduino-Programm

Das Programm des Arduino empfängt die beiden 4-Byte-codierten Koordinatenwerte und entschlüsselt diese zurück in Integer-Werte. Diese Werte dienen dann als Istwerte für die PID-Regelung. Die Sollwerte sind 320 (X) sowie 240 (Y) und repräsentieren den Mittelpunkt des Kamerabildes. Als Ausgangswert für das Digital-Potentiometer wird 127 auf beiden Kanälen initialisiert. Dies entspricht der neutralen Mittelstellung der Analogsticks auf der Fernbedienung. Für die PID-Regelung gibt es eine Library, welche die Berechnung der Ausgangswerte übernimmt.

```
#include <PID_v1.h>
#include <SPI.h>


const int ChipSelect      = 8;         //Chip-Select am Pin 8 des Arduino
const int AnalogPin       = A0;        //Liest den aktuellen Volt-Wert

const int wiper0writeAddr = B00000000; //Schreibadresse für Poti 0 des Digi-Pot
const int wiper1writeAddr = B00010000; //Schreibadresse für Poti 1 des Digi-Pot

const int led             = 9;    //Pin 9: LED

int x_wert                = 320;  //Wert der X-Koordinate
int y_wert                = 240;  //Wert der Y-Koordinate
int digipot_wert          = 127;  //Wert zwischen 0 und 255 für das Digital-Potentiometer

String inString           = "";   //String für die Serial-Kommunikation

const int printing        = 1;    //Aktiviert oder deaktviert die Serial-Ausgabe


/* ############### PID ##################*/

//Definiert die Variablen für den PID-Regler
double SetpointX, InputX, OutputX;
double SetpointY, InputY, OutputY;

//Die Einstellung der Werte des P-, I- und D-Anteils
double Kp=0.65,
       Ki=0.02,
       Kd=0.18;

//Initialisiert die Regler für die X- und Y-Koordinate
PID PIDX(&InputX, &OutputX, &SetpointX, Kp, Ki, Kd, DIRECT);
PID PIDY(&InputY, &OutputY, &SetpointY, Kp, Ki, Kd, DIRECT);
```

```
void setup() {

  // Setzt Chip-Select als AUSGANG und deaktviert ihn:
  pinMode(ChipSelect, OUTPUT);
  pinMode(ChipSelect, HIGH);

  // Initialisiert SPI und Serial:
  SPI.begin();
  Serial.begin(38400);
  Serial.println("Starte Programm\n");

  //LED als AUSGANG
  pinMode(led, OUTPUT);

  //Initialisiert beide Kanäle des Digital-Poti auf den Wert 127
  PotiSteuerung(127, 0);
  PotiSteuerung(127, 1);

  //Ausgangswerte für den Regler
  InputX = 320;
  InputY = 240;

  SetpointX = 320;
  SetpointY = 240;

  OutputX = 127;
  OutputY = 127;

  PIDX.SetMode(AUTOMATIC);
  PIDY.SetMode(AUTOMATIC);

}

void loop() {

  while (Serial.available() > 0) {
  SerialVerbindung();
  }

  InputX = x_wert;
  InputY = y_wert;

  PIDX.SetTunings(Kp, Ki, Kd);
  PIDY.SetTunings(Kp, Ki, Kd);

  PIDX.Compute();
  PIDY.Compute();

  PotiSteuerung(OutputX, 0);
  PotiSteuerung(OutputY, 1);

  Print();

  LedSteuerung();
}

void PotiSteuerung(int digipot_wert, int poti01){

  //Sendet den Einstellwert an den jeweiligen Kanal des Digital-Poti

  int adresse;
  if(poti01 == 0){adresse = wiper0writeAddr;}
  if(poti01 == 1){adresse = wiper1writeAddr;}

  digitalWrite(ChipSelect,LOW);
  SPI.transfer(adresse);
  SPI.transfer(digipot_wert);
  digitalWrite(ChipSelect,HIGH);
}
```

```
void SerialVerbindung(){

    //Empfängt die X- und Y-Werte des getrackten Objektes im Kamerabild

    int inChar = Serial.read();
    if (isDigit(inChar)) {
      inString += (char)inChar;
    }

    if(inChar == 42) {

      x_wert = inString.toInt();

      Print();
      inString = "";
    }

    if (inChar == 43) {

      y_wert = inString.toInt();

      Print();
      inString = "";
    }

}

void LedSteuerung(){

  //LED wird aktiviert, wenn die Abstände beider Achsen nahe am Sollwert liegen

  double abstandX = abs(SetpointX-InputX);
  double abstandY = abs(SetpointY-InputY);

  if(abstandX < 30 && abstandY < 30){digitalWrite(led, HIGH);}
  else{digitalWrite(led, LOW);}

}

void Print(){
  if(printing == 1){

  // Gibt die Werte auf der Konsole aus

  Serial.print("X-Wert: ");
  Serial.print(x_wert);

  Serial.print("\t OutputX: ");
  Serial.print(OutputX);

  Serial.print("\t Y-Wert: ");
  Serial.print(y_wert);

  Serial.print("\t OutputY: ");
  Serial.print(OutputY);

  float analog = analogRead(AnalogPin);
  Serial.print("\t Voltage: ");
  Serial.println(analog/204.6);
  }
}
```