Computer vision

De link tussen VR en fysieke wereld

*Pieter-Jan Cassiman*

Voorwoord

Inhoud

[1. Inleiding 8](#_Toc7963388)

[2. MQTT 9](#_Toc7963389)

[2.1. Principe 9](#_Toc7963390)

[2.2. Protocol 9](#_Toc7963391)

[2.3. Broker 9](#_Toc7963392)

[3. Controllers 10](#_Toc7963393)

[3.1. Joystick 10](#_Toc7963394)

[3.1.1. Ontwerp 10](#_Toc7963395)

[3.1.2. Functie 10](#_Toc7963396)

[3.1.3. Programma 10](#_Toc7963397)

[3.2. Gamepad 10](#_Toc7963398)

[3.2.1. Ontwerp 10](#_Toc7963399)

[3.2.2. Functie 10](#_Toc7963400)

[3.2.3. Programma 10](#_Toc7963401)

[3.3. Andere mogelijkheden 10](#_Toc7963402)

[4. Autootje 11](#_Toc7963403)

[4.1. Werkingsprincipe 11](#_Toc7963404)

[4.2. Ontwerp 11](#_Toc7963405)

[4.3. Functies 11](#_Toc7963406)

[4.3.1. Ingebouwd 11](#_Toc7963407)

[4.3.2. Uitbreiding 11](#_Toc7963408)

[4.4. Programma 11](#_Toc7963409)

[5. Bord 12](#_Toc7963410)

[5.1. Opbouw 12](#_Toc7963411)

[5.2. Muurtjes 12](#_Toc7963412)

[5.3. RGB matrix 12](#_Toc7963413)

[5.3.1. Schema 12](#_Toc7963414)

[5.3.2. Programma 12](#_Toc7963415)

[6. Vision 13](#_Toc7963416)

[6.1. Camera 13](#_Toc7963417)

[6.2. Systeem 13](#_Toc7963418)

[6.3. Framework 14](#_Toc7963419)

[6.3.1. Programmeertaal 14](#_Toc7963420)

[6.3.2. Bibliotheken 15](#_Toc7963421)

[6.3.3. Programma 16](#_Toc7963422)

[7. Linefollower 29](#_Toc7963423)

[7.1. Programma 29](#_Toc7963424)

[8. Bestuurbaar autootje 30](#_Toc7963425)

[9. Toetsenbord + MQTT echo 31](#_Toc7963426)

[10. Makerfaire Gent 2019 bevindingen 32](#_Toc7963427)

[10.1. Opstelling 32](#_Toc7963428)

[10.2. Opmerkingen 32](#_Toc7963429)

[11. Verdere mogelijkheden 34](#_Toc7963430)

[12. Besluit 35](#_Toc7963431)

[12.1. Toekomst visie 35](#_Toc7963432)

[12.2. Resultaten 35](#_Toc7963433)

[12.3. Wat moet er nog gebeuren 35](#_Toc7963434)

[13. Extra 36](#_Toc7963435)

[13.1. PID 36](#_Toc7963436)

[13.1.1. Proportioneel – P 36](#_Toc7963437)

[13.1.2. Integraal – I 37](#_Toc7963438)

[13.1.3. Afgeleide – D 37](#_Toc7963439)

[13.1.4. Continue PID 38](#_Toc7963440)

[13.1.5. Discrete PID 38](#_Toc7963441)

[13.2. Vision 38](#_Toc7963442)

[13.2.1. Kernel convolution 38](#_Toc7963443)

[13.2.2. Blur 38](#_Toc7963444)

[13.2.3. Edge detection 38](#_Toc7963445)

[Figuren 39](#_Toc7963446)

[Code fragmenten 40](#_Toc7963447)

[Tabellen 41](#_Toc7963448)

[Literatuurlijst 42](#_Toc7963449)

[Index 43](#_Toc7963450)

[Bijlagen 44](#_Toc7963451)

Samenvatting

Overzicht volledige systeem

Vision

De vision wordt volledig uitgelegd onder Vision op p13.

# Inleiding

Samen met mr. Sierens wouden we de praktische kant van onze opleiding verbeteren. Dit door de inhoud van de verschillende vakken beter op elkaar en het werkveld af te stemmen. Maar ook door, vanaf de eerste dag, aan te tonen waarom de inhoud nuttig is, en waar het later zal terugkomen. Vergelijkbaar met STEM-onderwijs.

Dit resulteerde in een project waar een combinatie van virtual-reality, fysieke objecten en abstractie gebruikt wordt. Dit zou de studenten moeten toelaten om een beter idee te krijgen van wat het zal zijn in het werkveld, vanaf de eerste dag van de opleiding.

Door het werken met een virtuele omgeving, kunnen (industriële) machines en installaties weergegeven en gesimuleerd worden. Waardoor de studenten een echte machine kunnen programmeren, in plaats van de klassieke LED’s. Hierdoor krijgen ze meer voeling met de realiteit.

Echter is het ook belangrijk dat de studenten nog steeds iets fysiek hebben om vast te houden. Vandaar dat eer een link nodig is tussen de virtuele omgeving en de fysieke wereld. Het belangrijkste aspect van deze link is het nauwkeurig kunnen positioneren van fysieke objecten, om deze dan over te brengen naar de virtuele omgeving.

Voor het lokaliseren van objecten werd gekozen voor een computer vision systeem. Dit is zowel veelzijdig en gemakkelijk te implementeren. Het is ook nauwkeurig genoeg om te detecteren op een klein oppervlak. Hierdoor blijft de hele opstelling passen op de doorsnee labotafel.

Voor onze bachelorproef hebben we ons voornamelijk gefocust op het computer-vision systeem en het kijken of dit een geschikte optie is voor het positioneren op een kleine oppervlakte. We hebben dit uitgewerkt als een proof-of-concept door autootjes te volgen langsheen een parcours en de rondetijden op te meten.

# MQTT

## Principe

Waarom werd er met MQTT gewerkt?

MQTT is een protocol om berichten en data te versturen over het netwerk. MQTT wordt gecontroleerd door een broker. Alle clients moeten enkel hun data versturen naar de broker. Bijgevolg moeten ze enkel het IP-adres van de broker kennen en onthouden. Elke client kan dan ook vragen aan de broker of er nieuwe berichten zijn.

MQTT is ondersteunt op enorm veel platformen en in zeer veel programmeertalen. Waardoor het zeer universeel is. Alle deze factoren samen genomen zorgen er voor dat het systeem zeer universeel is en aan te sturen via verschillende platformen.

MQTT zou ook de gebruikte interface worden tussen de virtual reality en alle dingen die de virtual reality willen aansturen en die bestuurd worden door de virtual reality. Om deze reden werd een deel geïmplementeerd met MQTT. Op deze manier was aan te tonen dat de vision gelinkt kan worden aan verschillende dingen. En dat het ook kan gebruikt worden om de resultaten van de vision door te sturen.

## Protocol

Voor het versturen van berichten over MQTT hebben we een eigen protocol opgesteld. Hierdoor konden de verschillende systemen data uitwisselen.

Het protocol verstuurt de berichten als een array van bytes. Iedere byte heeft zijn eigen betekenis. Op deze manier kan de data snel een efficiënt uitgewisseld worden. Maar kunnen de verschillende systemen toch nog een onderscheid maken tussen welke berichten voor hen zijn en welke niet.

Door de structuur van het protocol kunnen de systemen ook zien wat ze moeten doen met de data die binnenkomt. Zo is er een byte die aanduidt voor welke functie de data bedoeld is. Een andere byte geeft het kanaal weer waarop de data van toepassing is.

Vergelijkbaar met de registers in een chip die geadresseerd worden. De functie en kanaal byte kunnen vergeleken worden met een inwendig adres waar de data naartoe moet.

## Broker

# Controllers

## Joystick

### Ontwerp

### Functie

### Programma

## Gamepad

### Ontwerp

### Functie

### Programma

## Andere mogelijkheden

# Autootje

## Werkingsprincipe

## Ontwerp

## Functies

### Ingebouwd

### Uitbreiding

## Programma

# Bord

## Opbouw

## Muurtjes

## RGB matrix

### Schema

### Programma

# Vision

“For both biological systems and machines, vision begins with a large and unwieldy array of measurements of the amount of light reflected from surfaces in the environment. The goal of vision is to recover physical properties of objects in the scene, such as the location of object boundaries and the structure, color and texture of object surfaces, from the two-dimensional image that is projected onto the eye or camera. This goal is not achieved in a single step; vision proceeds in stages, with each stage producing increasingly more useful descriptions of the image and then the scene. The first clues about the physical properties of the scene are provided by the changes of intensity in the image” (Hildreth, 1985)

## Camera

Zo goed als elke camera kan gebruikt worden voor het vision systeem. Van de simpelste webcam, tot industriële CMOS camera’s. De camera module van Raspberry Pi zit daar ook tussen.

Het vision systeem werd ontwikkeld en getest met een Logitech USB-webcam op een laptop, echter kan dit gemakkelijk overgezet worden naar de Raspberry Pi. Na het downloaden en installeren van de benodigde bibliotheken kan het programma zonder problemen overgezet worden naar de Raspberry Pi. Op de benodigde bibliotheken wordt verder teruggekomen.

Door het stuk code aan te passen dat de webcam start, kan de USB webcam vervangen worden door de Raspberry Pi camera module. Voor de rest van het programma verandert er niets. Hier wordt verder op ingegaan in Frame inlezen pagina 17.

## Systeem

Door het gebruik van Python kan het vision systeem op, zo goed als, elk systeem draaien.

Echter moet dit genuanceerd worden. Elk programma dat beeldherkenning of beeldverwerking doet, is een relatief zwaar programma. Bijgevolg zijn er wel enkele minimum vereisten voor het systeem:

|  |  |
| --- | --- |
| Specificatie | Vereiste |
| RAM | Minstens 1 GB (2 GB+ aangeraden) |
| CPU-kernen | Minstens 2 |
| CPU-snelheid | Minstens 1.2 GHz |
| Opslag | Minstens 8 GB |

Tabel 6.1: Minimum vereisten voor het vision systeem

De Raspberry Pi 3B valt nog net binnen deze vereisten. Het deel van het programma, dat verantwoordelijk is voor de vision, werd getest op een Raspberry Pi 3B. Het programma draaide en de vision werkte, echter werd de Raspberry Pi zeer warm. Doordat de visualisatie een beetje aan de wensen overlaat, werd er verder gewerkt met een laptop. Een Raspberry Pi 3B+ zou misschien beter geweest zijn, aangezien deze meer RAM heeft en een betere koeling voor de processor.

Het programma draaien op een moderne desktop of laptop zou geen enkel probleem mogen vormen, aangezien deze meer dan genoeg middelen hebben.

## Framework

### Programmeertaal

De gebruikte programmeertaal voor het vision systeem is Python 3, specifiek Python 3.7.2 64bit.

“Python is powerful... and fast;

plays well with others;

runs everywhere;

is friendly & easy to learn;

is Open.” (The Python Software Foundation)

Python is een enorm populaire programmeertaal, zeker in de wetenschappelijke wereld. Bijgevolg zijn er enorm veel modules en uitbreiding beschikbaar voor Python. Voor bijna elke toepassing is er minstens één module beschikbaar voor Python.

Deze taal werd gekozen om verschillende redenen. De grootste reden was de uitgebreide community en documentatie rond Python. Verder woog onze uitgebreide voorkennis van Python ook zwaar door. Python programmeert en leest zeer gemakkelijk. Hierdoor kunnen derden achteraf ook zeer gemakkelijk wijs geraken uit de code en hun eigen aanpassingen aanbrengen.

Door de object-georiënteerde aanpak van Python kan de code ook zeer gemakkelijk opgesplitst worden in verschillende delen die elk verantwoordelijk zijn voor een klein onderdeel van het geheel. Deze opsplitsing vereenvoudigt het lezen en werken verder.

PyPi (Python Package Index) is de module manager van python, deze bevat veel van de modules die beschikbaar zijn. Via PIP (Package Installer for Python) zijn deze modules zeer gemakkelijk te installeren.

### Bibliotheken

#### OpenCV

Voor het inlezen en daadwerkelijk verwerken van de beelden wordt OpenCV (Open Source Computer Vision Library) gebruikt. Dit is een bibliotheek voor beeldherkenning en –verwerking. OpenCV is een enorm populaire bibliotheek, die gebruikt word door bedrijven als Google, Microsoft, Intel, IBM en Sony. Enkele van de toepassingen van OpenCV zijn: het samenvoegen van de afbeelding voor Streetview, het monitoren van mijn gereedschap en machines in China, Het detecteren van verdrinkingen in zwembaden, startbanen controleren op puin, etc.

OpenCV kan gebruikt worden in C++, Python, Java en MATLAB op verschillende platformen, waaronder Windows, Linux en Android.

OpenCV zelf is geschreven in C++, en moet gecompileerd worden. Echter is er op PyPi een voor-gecompileerde versie te vinden die via PIP zeer gemakkelijk te installeren is. Dit is de versie die gebruikt werd in ons programma (versie 4.0.0 op PyPi).

Voor het installeren van OpenCV kan in een terminal venster het volgende commando ingevoerd worden:



Code 6.1: PIP commando voor het installeren van OpenCV

Dit zal de laatst nieuwe versie installeren. Om een specifieke versie te installeren kan een commando opgeroepen worden als volgt:



Code 6.2: PIP commando voor het installeren van OpenCV 4.0.0

Er zijn echter verschillende versies van OpenCV beschikbaar voor Python:

* opencv-python
* opencv-contrib-python
* opencv-python-headless
* opencv-contrib-python-headless

De headless versies zijn enkel te gebruiken in het programma, en hebben geen enkele visuele functionaliteit. Dit wil zeggen dat de headless versies gebruikt kunnen worden zoals normaal, enkel kan het resultaat niet weergegeven worden op het scherm, vanuit het programma. Aangezien OpenCV ook gebruikt wordt voor de visualisatie, is het best om deze versies niet te installeren.

De contrib versies bevatten, naast de standaard functionaliteit, ook uitbreidingen die aangebracht zijn door de community. Deze versies hebben uiteraard veel meer functionaliteit, echter is de installatie grootte navenhand.

Voor deze toepassing is de gewone versie voldoende (opencv-python). Deze kan geïnstalleerd worden met de eerder vermelde commando’s.

#### NumPy

NumPy is een enorm krachtige module, die veel gebruik word in wetenschappelijke context. Het meest gekende (en grootste) aspect van NumPy, zijn de N-dimensionele arrays. Deze arrays en bijhorende operaties zijn enorm efficiënt geïmplementeerd waardoor deze bewerkingen zeer snel uitgevoerd worden ook met grotere arrays.

Er zijn enorm veel andere Python modules die NumPy gebruiken voor het implementeren van het datastructuren. Hier wordt NumPy gebruikt om het array van het frame op te slaan. De resultaten zijn telkens ook een NumPy array, evenals de gedetecteerde contouren.

#### Imutils

Imutils is een relatief kleine module voor Python. Met deze module is het gemakkelijker om afbeeldingen te verplaatsen, draaien, vergroten en te verkleinen. Deze module heeft ook een functie voor het omvormen en sorteren van contouren, gegeven door OpenCV. Het is deze functionaliteit die gebruikt wordt in dit programma.

### Programma

Nieuwe volgorde, webcam object etc..:

Het volledige programma is terug te vinden in bijlage, hier wordt specifiek ingegaan op het deel verantwoordelijk voor de vision.

#### Bibliotheken importeren

Als eerste moeten de benodigde bibliotheken geïmporteerd worden. Er zijn drie bibliotheken die in ieder geval nodig zijn; OpenCV, Imutils en NumPy



Code 6.3: Importeren van de bibliotheken

Als er gewerkt wordt met de Raspberry Pi camera, zijn er nog enkele bibliotheken die extra nodig zijn. Deze voorzien de correcte werking en interface om de camera module te kunnen gebruiken met OpenCV.



Code 6.4: Importeren van de bibliotheken voor de Raspberry Pi camera

#### Frame inlezen

Aangezien er gewerkt wordt met live-video, moet er aan het begin van elke programma lus een nieuw frame ingelezen worden. Samen met de opstart van de camera zijn dit de enige stukken code die (kunnen) verschillen afhankelijk van de camera. Ze zullen hier dus ook samen bekeken worden.



Code 6.5: Code voor het gebruik van een USB camera

Om te werken met een USB camera, moet er eerst een VideoCapture object aangemaakt worden. De constructor voor dit object krijgt één argument mee. Dit argument geeft als het ware het kanaal van de camera mee. Op deze manier kunnen meerdere camera’s verbonden worden met dezelfde computer en gebruikt worden in OpenCV. OpenCV kan dus ook gebruikt worden voor 3D toepassingen met twee camera’s.

Op een desktop met slechts één camera aangesloten zal het kanaal altijd 0 zijn. Op een laptop, met een ingebouwde camera, kan het zijn dat deze camera als kanaal 0 wordt aanzien en dat een webcam dus kanaal 1 zal krijgen. De laptop waarop het programma getest werd had een ingebouwde camera, echter was de USB camera op kanaal 0 te vinden.

Voor het gebruik van de USB camera moest geen enkele driver geïnstalleerd worden. Echter hebben werd slechts één USB camera getest, dus het is mogelijk dat voor andere camera’s wel drivers geïnstalleerd moeten worden.

Echter wordt de webcam nu in een oneindige lus uitgelezen, het programma is dus enkel te stoppen via een KeyboardInterrupt (ctrl + c), dit is een geforceerde manier om het programma te stoppen.

Echter kan het scherm, dat door OpenCV getoond wordt, gebruikt worden om het programma gecontroleerd af te sluiten. Zolang het venster open is, kan de webcam verder uitgelezen worden. Op het moment dat het visualisatie scherm gesloten wordt, door bv. het windows-kruisje in te duwen, moet het programma stoppen. Dit wordt geïmplementeerd door de while-lus te vervangen door het volgende fragment:



Code 6.6: While lus voor het kunnen sluiten van het programma

Echter kan het programma op een andere, gecontroleerde, manier afgesloten worden. Het programma kan ook gestopt worden door een toets in te duwen. Dit wordt verkregen door het volgende fragment:



Code 6.7: Code voor het stoppen van OpenCV met een toets

Voor het gebruik van de Raspberry Pi camera, zijn twee bijkomende bibliotheken nodig.



Code 6.8: Code voor het gebruik van de Raspberry Pi Camera module

Met de eerste bibliotheek, PiCamera, kan de camera module gebruikt en bestuurd worden op de Raspberry Pi via Python. Deze bibliotheek kan aanzien worden als een driver.

De tweede bibliotheek, PiRGBArray, dient enkel voor het correct “formateren” van de data die van de camera module komt. Door de data correct te formateren kan de camera module gebruikt worden in OpenCV.

Net als bij de USB-webcam moet een camera object aangemaakt worden. Dit object kan dan gebruikt worden voor het instellen van de vereiste parameters, zoals de resolutie[[1]](#footnote-1) en de framerate. Vervolgens kan een object aangemaakt worden dat de video stream van de camera module beschikbaar maakt voor de rest van het programma.

De while-lus is nu vervangen door een for-lus. Beide hebben hetzelfde effect, bij de start van elke lus wordt een frame ingelezen en verwerkt.

Echter is op de Raspberry Pi het programma enkel te stoppen via een toets.

Als alles gelukt is, wordt het volgende beeld verkregen, dit wordt verder verwerkt in het programma.



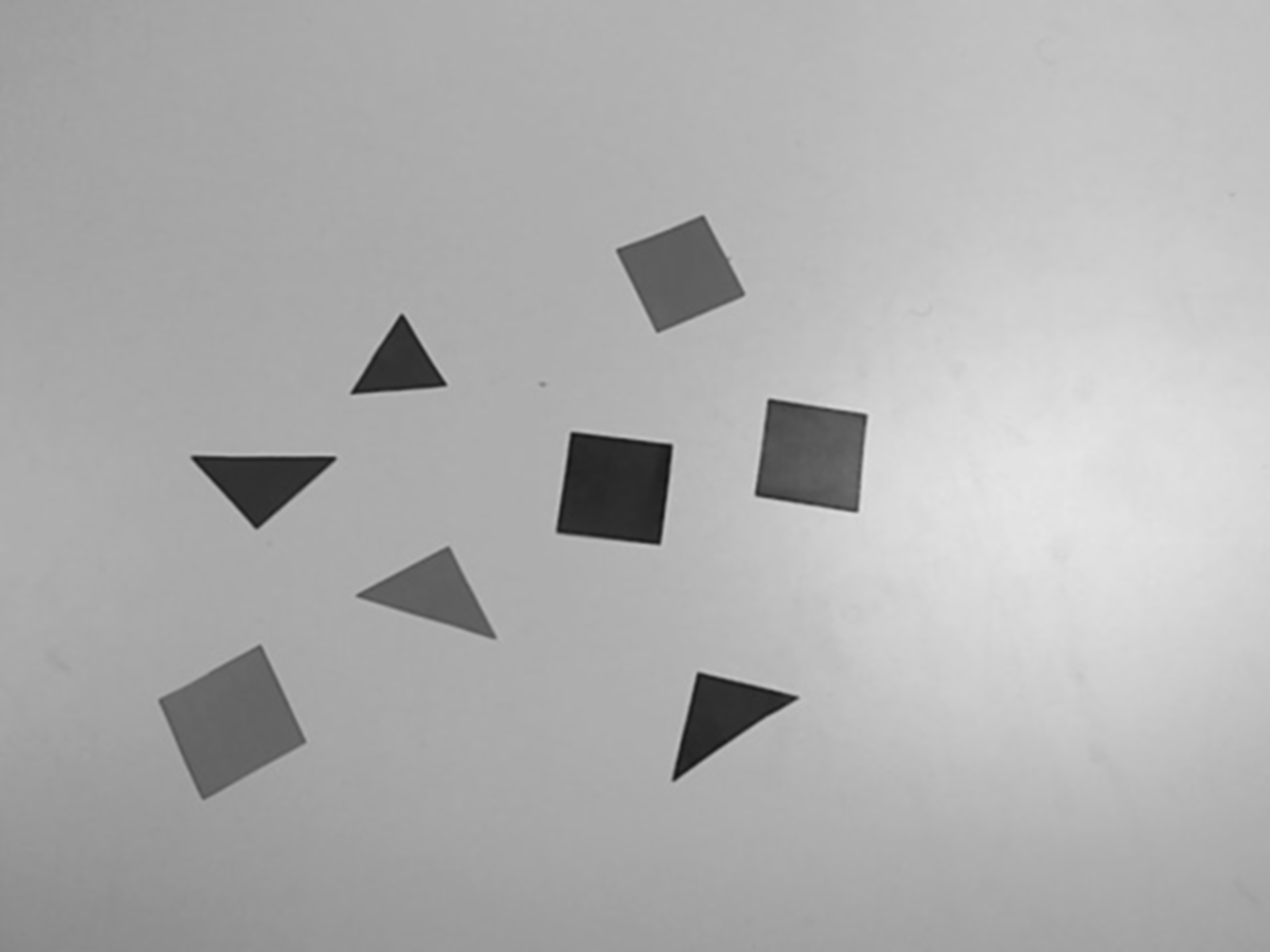
Figuur 6.1: Het ingelezen frame

#### Omzetten naar grijstinten

De allereerste bewerking die wordt uitgevoerd op het ingelezen frame, is een omzetting naar grijstinten. Een kleurenbeeld bevat drie kanalen per pixel (rood, groen en blauw). Voor de bewerkingen die volgen worden alle pixels in het hele frame overlopen en worden de bewerkingen uitgevoerd op alle kanalen van alle pixels.

Door het omzetten naar grijstinten worden de drie kanalen herleidt tot in kanaal dat de intensiteit weergeeft. Hierdoor bedraagt het aantal benodigde berekeningen slechts een derde, in vergelijking met een kleurenbeeld. Waardoor de verwerking ook, ruwweg, drie keer zo snel gebeurt.

Echter wordt er wel een kopie bewaard van het kleurenbeeld, zodat dit later kan weergegeven worden. Het kleurenbeeld wordt ook gebruikt om later de kleur te bepalen van de gedetecteerde vormen.



Figuur 6.2: Frame omgezet naar grijstinten

Het omzetten naar grijstinten wordt als volgt verkregen in OpenCV:



Code 6.9: Code fragment kleur omzetten naar grijstinten

#### Gaussian blur

De volgende operatie is een Gaussian blur, dit om de hoeveelheid ruis te verminderen. De gebruikte kernel werd proefondervindelijk bepaald. De grootte is een balans tussen het onderdrukken van ruis en het behouden van detail.

In OpenCV wordt een Gaussian blur als volgt verkregen:



Code 6.10: Code fragment Gaussian blur en kernel

De parameters die aan deze functie moeten worden meegegeven zijn als volgt:

Het beeld waarop de operatie moet worden uitgevoerd, de kernel en de standaardafwijking. Als de standaardafwijking nul is, wordt deze berekend uit de kernel.

Meer detail en uitleg over Gaussian blur en bijhorende parameters, kan gevonden worden onder Gaussian blur op pagina 35.

#### Threshold

Aangezien het hier gaat over het detecteren van vormen, is een zo hoog mogelijk contrast wenselijk. Het contrast kan verhoogd worden tot er slechts twee mogelijke waardes over zijn voor een pixel. Dit kan bekomen worden op verschillende manieren, de meest efficiënte manier is een binaire drempel (binary threshold).

Hierbij wordt elke pixel vergeleken met een drempel waarde, is de pixel minder dan de drempel waarde, dan wordt hij zwart gezet. Is de pixel hoger dan de drempel waarde, dan wordt hij wit gezet.

Bijgevolg wordt een zoor hoog contrast verkregen, dat zeer geschikt is voor verdere operaties. De drempel waarde werd ook proefondervindelijk bepaald.

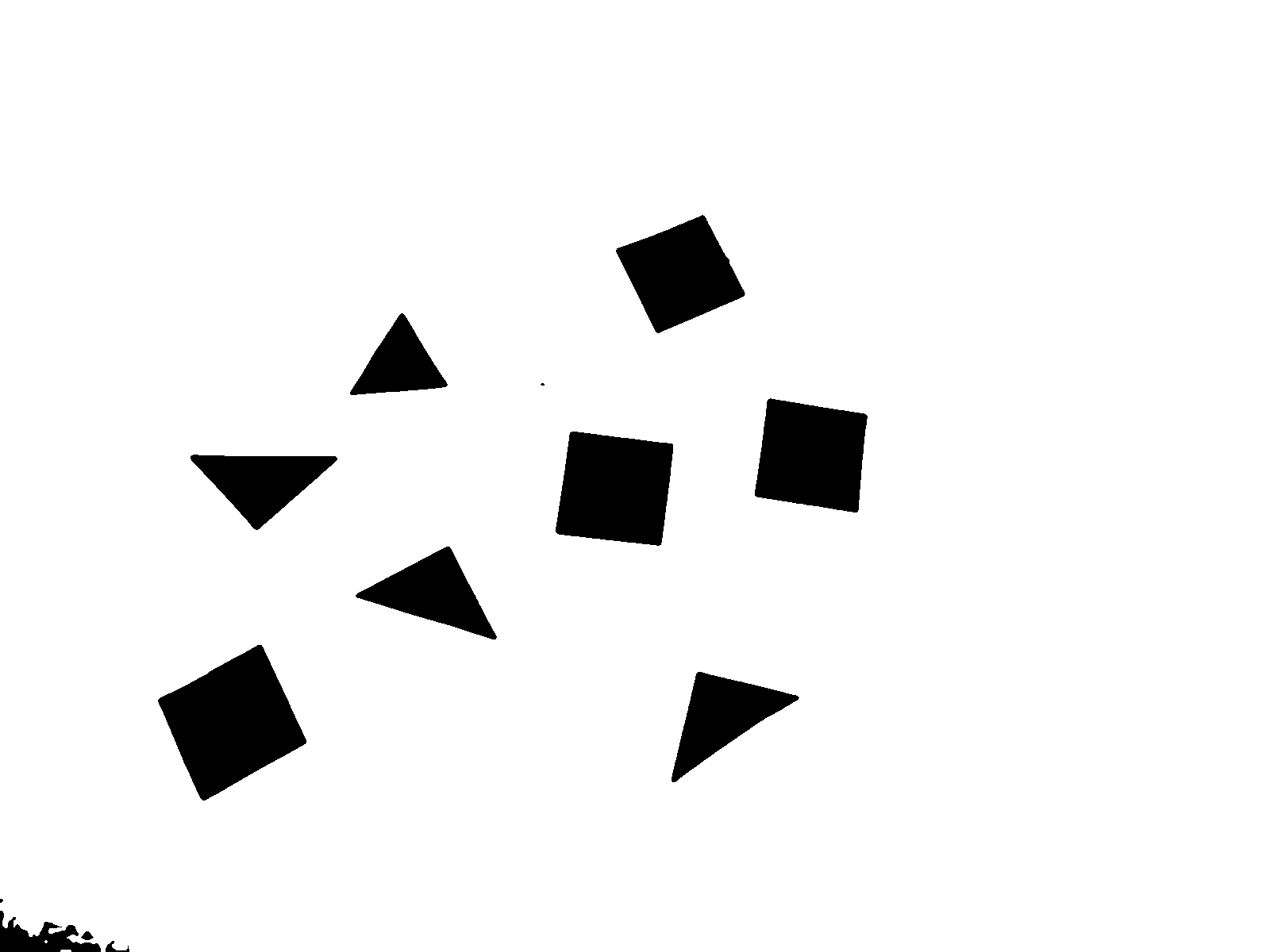
In OpenCV kan een threshold verkregen worden als volgt:



Code 6.11: Code fragment threshold

De functie geeft twee waardes terug, de eerste waarde die teruggegeven wordt is de drempelwaarde. Deze is echter van toepassing voor andere drempel technieken. De teruggegeven drempelwaarde moet dus niet bijgehouden worden (voorgesteld door de “\_”). De tweede waarde die teruggegeven wordt is het resultaat van de threshold operatie. Deze wordt opgeslagen als een nieuwe afbeelding.

Er moeten een aantal parameters meegegeven worden aan de functie; van links naar rechts: Het originele beeld waarop de operatie wordt uitgevoerd, de drempelwaarde, de maximumwaarde en het type threshold dat gebruikt wordt. De maximumwaarde is de “witte” waarde voor de pixel, hier ingesteld op 255.



Figuur 6.3: Resultaat van de threshold operatie

Op het resultaat is te zien hoe alle figuren omgevormd zijn tot volledig zwarte vlakken op een witte achtergrond, door de drempel operatie. Echter is er ook nog een restant van de ruis gedetecteerd. Deze ruis vormt geen direct probleem, deze wordt bij latere operaties verder uitgefilterd.

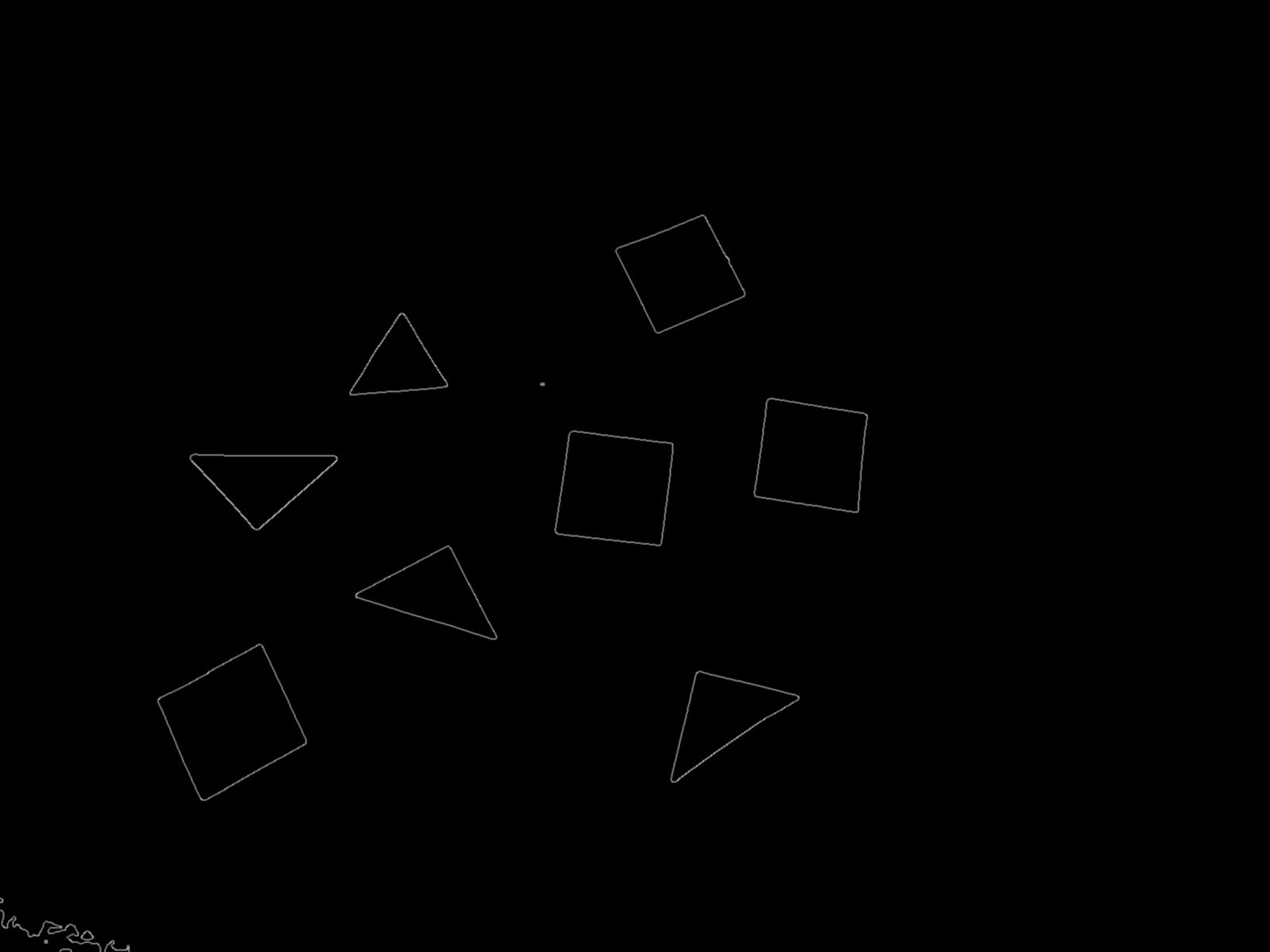
Voor de edge-detectie is normaal het inverse vereist. Witte vormen op een zwarte achtergrond. In OpenCV is een afbeelding zeer gemakkelijk te inverteren. Echter werden niet de gewenste resultaten verkregen met een invers beeld. Vandaar dat er werd verder gewerkt met het resultaat van de drempel operatie, zonder dit de inverteren.

#### Canny edge detection

De figuren zijn herleidt tot volle vlakken, echter kunnen ze nog altijd niet herkend worden door het programma. Een mens kan kijken en onmiddellijk zien welke figuur overeenkomt met welke vorm. Een mens doet dit uit instinct, maar hoe wordt een vorm herkend? Vier zijdes betekent een vierkant (of rechthoek), drie zijdes betekent een driehoek, enzoverder. Dit is exact dezelfde manier waarop de computer de vormen herkend.

De volgende stap is het detecteren van de zijdes. Dit wordt verkregen door het detecteren van randen in het beeld. Voor het detecteren van de randen wordt Canny edge detection gebruikt.

Door alle eerder uitgevoerde operaties is het beeld ondertussen veel vereenvoudigd waardoor enkel de randen (en omtrek) van de figuren bekomen wordt. Er is opnieuw ruis dat gedetecteerd wordt.



Figuur 6.4: Resultaat van edge-detectie

Op de gedetecteerde randen wordt nogmaals een Gaussian blur gebruikt, om verder ruis te onderdrukken. Hierdoor worden ook kleine imperfecties in de randen onderdrukt waardoor het detecteren van de contouren veel beter gaat. Hierdoor worden de vormen nauwkeuriger gedetecteerd. In de volgende stap wordt hier verder op ingegaan.

Het detecteren van de randen in OpenCV gebeurt als volgt:



Code 6.12: Canny edge detection in OpenCV

Meer detail over de edge detectie kan gevonden worden onder Edge detection op pagina 31.

#### Contouren

Via volgend code fragment worden de contouren uit het resultaat van de edge detection gehaald, en geformatteerd.



Code 6.13: Contouren detecteren en isoleren in OpenCV

De laatste lijn van dit fragment formatteert de gevonden contouren als een lijst. Hierdoor kunnen de gevonden contouren gemakkelijk overlopen worden in een for-lus.



Code 6.14: For-lus om over de contouren te lopen en te filteren

Voor iedere gedetecteerde contour wordt de omtrek berekend (arc lengte). Deze wordt dan gebruikt om verder ruis weg te filteren. Te kleine vormen worden genegeerd.

Als laatste wordt een veelhoek benaderd op basis van de contour en de omtrek. Deze laatste wordt dan gebruikt voor het identificeren en lokaliseren van figuren.

#### Hoekpunten tellen in de contour & vorm bepalen

De vorm wordt bepaald aan de hand van het aantal hoekpunten dat te vinden is in een contour. Door het uitvoeren van een tweede Gaussian blur, na het detecteren van de randen, worden imperfecties in de randen onderdrukt. Hierdoor worden de contouren als het waren verzacht, of afgerond. Door dit de doen verliezen we een deel nauwkeurigheid, maar worden extra hoekpunten onderdrukt. Deze extra hoekpunten kunnen ontstaan doordat de contour niet perfect is, of omdat de vormen niet perfect uitgeknipt zijn. Ook kan het zijn dat de hoekpunten op het fysieke object afgesleten zijn.

Voor iedere contour worden de hoekpunten geteld, de soort figuur wordt bepaald aan de hand daarvan.



Code 6.15: Hoekpunten differentiëren in de contouren

Python beschikt niet over een switch functie, echter kan ditzelfde resultaat bekomen worden via een if-else structuur. Er kan een onderscheid gemaakt worden tussen elk nummer van hoekpunten, echter op een bepaald punt is het onderscheid tussen de vormen te klein. Vandaar dat het aantal hoekpunten best beperkt wordt tot een maximum van acht.

#### Zwaartepunt van de vorm bepalen

#### Kleur bepalen

[eerst functie herschrijven dan uitleg schrijven]

#### Hoekpunten en kleur doorgeven naar de rest van het programma

[verdere uitleg]



Figuur 6.5: Resultaat van het vision-programma

# Linefollower

## Programma

# Bestuurbaar autootje

# Toetsenbord + MQTT echo

# Makerfaire Gent 2019 bevindingen

Als deel van onze stage en bachelorproef hebben we ons project gepresenteerd op de Makerfaire hier in Gent, we stonden er onder de naam van Makers for Education.

Dit was ook zeer goede kans om ons project en de verschillende systemen. De systemen hebben het zeer goed uitgehouden, echter kwamen er enkele bevindingen naar boven.

## Opstelling

De opstelling voor de Makerfaire bestond uit twee delen. Ook waren er twee demonstraties die getoond konden worden.

Het eerste deel van de opstelling bestond uit het bord met de webcam en het parcours voor de linefollower.

Het tweede deel waren de computers waarop de software draaide. Er was één computer die de vision bestuurde en een tweede computer die een toetsenbord had, dat de bezoekers konden gebruiken voor het besturen van een autootje.

De eerste demonstratie was een linefollower die autonoom de lijn kon volgen. De tweede demonstratie was een autootje dat de bezoekers konden besturen via een toetsenbord dat aan één van de computers hing. De toetsaanslagen werden op de computer ingelezen en dan via ons protocol doorgestuurd naar het autootje.

Beide autootjes werden langs het hele traject gevold door het vision systeem. Het vision systeem hield de rondetijden bij en gaf na drie rondes het gemiddelde en de snelste tijd weer.

Op deze manier konden we alle aspecten van onze bachelorproef testen en tonen.

Ook waren er een aantal aspecten van het systeem die ontworpen waren, maar die niet meegenomen waren naar de Makerfaire. Ondermeer de controllers die we ontworpen hadden. Deze werkten veel te onbetrouwbaar om te laten gebruiken door de bezoekers.

Ook waren de muurtjes en de RGB-matrix niet mee. De muurtjes hadden we niet nodig aangezien we er geen toepassing voor hadden. De RGB-matrix had dezelfde reden, ook was deze niet afgeraakt.

## Opmerkingen

* Vision enorm gevoelig voor veranderingen in omgevingslicht, net als de linefollower.
* Vision is zeer stabiel en betrouwbaar (los van de belichting). In een gecontroleerde omgeving zoals een klaslokaal of afgesloten ruimte, is dit dus zeer goed toe te passen. Ofwel is er een constante lichtbron nodig, ofwel een betrouwbare manier om de *exposure* van de camera automatisch te laten aanpassen.
* Het vision systeem draait zeer vlot via een webcam en een doorsnee computer. Dit is dus zeer geschikt voor locatie toepassingen op een zeer kleine oppervlakte.
* Voor MQTT is het best alles te verbinden via kabel (om vertragingen en storingen te voorkomen). Voor de rest is het ook wel gemakkelijk in gebruik.
* Een bestuurbaar autootje is enorm populair bij het jongere publiek, kan dus zeer goed gebruikt worden bij het voorstellen van de opleiding, zoals opendeurdagen etc. Of voor workshops in het lager / middelbaar. Dit autootje moet niets speciaal zijn, gewoon twee wielen, een kogellager, een controller en wifi of bluetooth.
* Een toetsenbord is misschien niet te beste optie voor de controller van het autootje.
* Arduino MEGA en wifi module is niet de beste keuze voor real-time toepassingen zoals het bestuurbaar autootje. De ESP8266 had minder last van vertragingen op de berichten. Een ESP32 zou nog beter zijn, gezien dat deze module meer I/O pinnen heeft en meer analoge ingangen.
* Door de vele aanwezige netwerken op de Makerfaire, hadden we problemen met het netwerk (voornamelijk doordat alle banden bezet waren en “overcrowded” waren).
* De batterijduur verraste ons, de batterijen hebben het de hele dag uitgehouden zonder te moeten vervangen worden. Ondanks dat de autootjes zo goed als constant gereden hebben.
* Voor publieke toepassingen zou het bord bet beschikken over een boord of rand zodat de autootjes er niet kunnen afrijden.
* De Makerfaire was een zeer goede *field test* voor onze bachelorproef.

# Verdere mogelijkheden

# Besluit

## Toekomst visie

## Resultaten

## Wat moet er nog gebeuren?

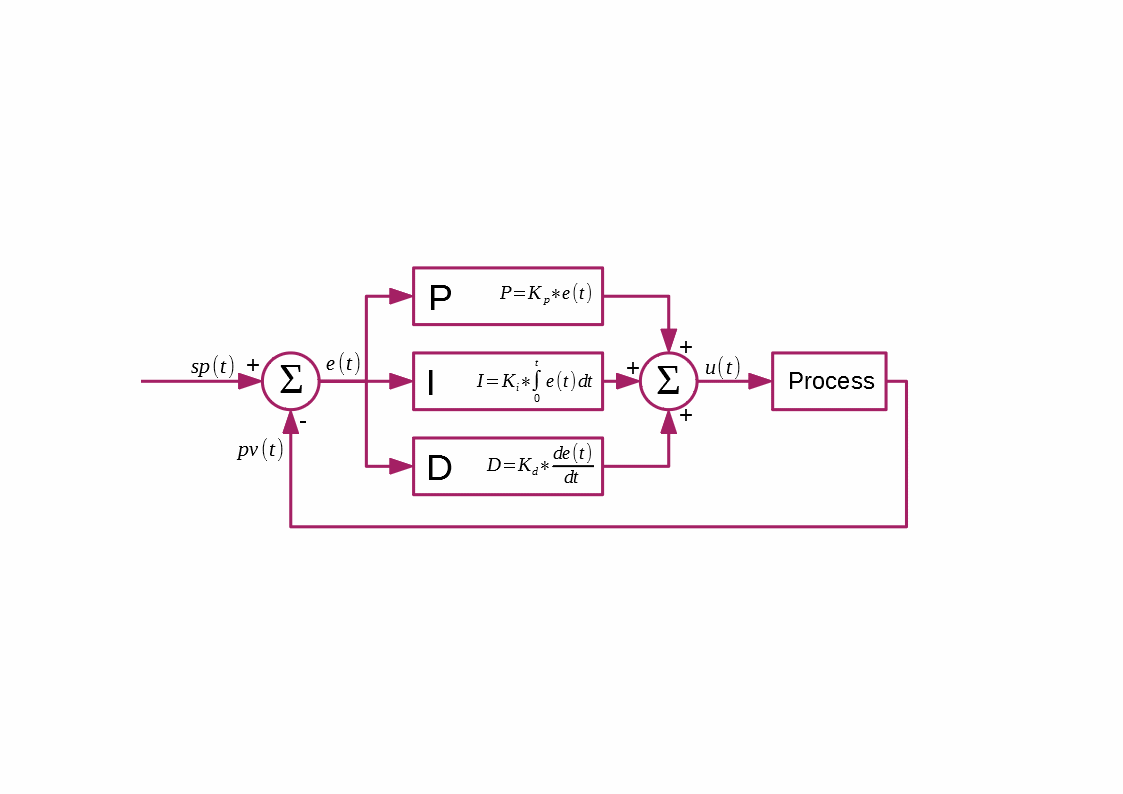
# Extra

## PID

Een PID regelaar is de populairste en meest gebruikte closed-loop controller.

Een PID regelaar krijgt een input (sp), dit is de gewenste waarde. Van deze waarde wordt de huidige sensorwaarde (pv), of proces waarde, afgetrokken. Het resultaat (e), of afwijking, is de input voor het eigenlijke PID algoritme. Dit algoritme bestaat uit drie parallelle functies; een Proportionele, een Integraal en een afgeleide (Derivative).

Het resultaat van elk van deze drie functies worden opgeteld (u), dit is dan ook de uitgang van de PID regelaar. Deze dient als input voor het proces. De uitgang van het proces (pv) is dan teruggekoppeld aan de PID regelaar, vandaar dat men spreekt over een closed-loop systeem.



Figuur 11.1: Schema van een PID regelaar

Elke van de functies in een PID regelaar heeft zijn eigen K-factor. Deze factor bepaald hoeveel elk van de functies doorwegen in het eindresultaat. Deze factoren worden ook gebruikt voor het tunen van de PID regelaar. Als deze factoren juist gekozen en ingesteld worden, kan een enorm snelle en nauwkeurige PID regelaar bekomen worden.

### Proportioneel – P

De eerste functie is tevens de meest eenvoudige. Zoals de naam het zegt, is de uitkomst van deze functie proportioneel aan de input. Kp zal bepalen in welke mate de uitgang proportioneel is aan de ingang. De formule is als volgt:

De P-functie is de meest eenvoudige, maar “kijkt” enkel naar de ogenblikkelijke afwijking. Hierdoor treed overshoot zeer snel op, waardoor een P-regelaar zal oscilleren. De D-functie is een oplossing hiervoor.

Ook zal bij zeer kleine afwijkingen de proportionele uitgang niet sterk genoeg zijn, om de kleine afwijking weg te werken. De integraal biedt hier een oplossing op.

### Integraal – I

De I-functie neemt de integraal van de afwijking. Bijgevolg zal de uitgang steeds groter en groter worden bij een constante kleine afwijking. De I-functie is uitermate geschikt voor het wegwerken van kleine afwijking die lang aanwezig zijn. De I-functie “kijkt” dus als het ware naar het verleden.

De I-functie wordt beschreven door volgende vergelijking:

De I-functie is voornamelijk van toepassing op systemen die langzaam veranderen. Bij systemen die snel veranderen zal de afwijking nooit lang genoeg aanwezig zijn om de uitgang van de I-functie noemenswaardig te doen stijgen en dus door te laten wegen.

De I-functie moet altijd begrensd worden. Als de regelaar een constante afwijking heeft, maar niet genoeg kan regelen om de afwijking weg te werken, zal de uitgang van de I-functie constant blijven groeien. In theorie kan deze dus oneindig groot worden.

In de praktijk is de uitkomst meestal beperkt zonder dat er iets voor gedaan moet worden. Als de regelaar analoog werkt, is de uitkomst van de I-functie beperkt tot de voedingsspanning. Als de regelaar digitaal werkt, zal er op een bepaald moment een overflow optreden. Dit is natuurlijk niet wenselijk aangezien de uitkomst van de I-functie dan van de grootste waarde plots naar de kleinste waarde gaat, of omgekeerd.

Hoe dan ook is de waarde beperkt. Echter is het best om eigen beperkingen in te stellen. Zodat ze voorspelbaar zijn en geen onverwacht gedrag met zee meebrengen.

### Afgeleide – D

De D-functie neemt de afgeleide van de afwijking. Deze kijkt naar hoe snel de afwijking veranderd. De D-functie “kijkt” dus naar de toekomst en kan dus beperkt de komende afwijking voorspellen.

De D-functie wordt gegeven door volgende vergelijking:

De D-functie zorgt dat de PID regelaar sneller reageert op veranderingen in de afwijking.

### Continue PID

In wiskundige, en theoretische, context is een PID regelaar altijd continu. De waardes veranderen vloeiend en continu.

### Discrete PID

## Vision

### Kernel convolution

Een kernel convolution is een wiskunde operatie waarbij een matrix over een andere, meestal veel grotere, matrix geleidt. De kleinere matrix wordt de kernel genoemd, deze heeft altijd een oneven grootte. De waardes van de kernel zijn meestal gebaseerd op een tweedimensionale normaalverdeling.

De kernel convolution wordt als volgt uitgevoerd;

1. De kernel wordt met het center over een pixel geplaatst.
2. Alle buren en pixel zelf worden vermenigvuldigd met de overeenkomstige waarde in de kernel
3. Alle bekomen waardes worden opgeteld en gedeeld door het totaal van de kernel.
4. Dit wordt gedaan voor elke pixel in de afbeelding
5. Zo wordt een nieuwe afbeelding bekomen.
6. Is equivalent aan een groot gewogen gemiddelde van iedere pixel in de afbeelding met zijn buren.

### Blur

#### Mean blur

#### Gaussian blur

### Edge detection

Figuren

[Figuur 6.1: Het ingelezen frame 21](#_Toc7637861)

[Figuur 6.2: Frame omgezet naar grijstinten 22](#_Toc7637862)

[Figuur 6.3: Resultaat van de threshold operatie 24](#_Toc7637863)

[Figuur 6.4: Resultaat van edge-detectie 25](#_Toc7637864)

[Figuur 6.5: Resultaat van het vision-programma 28](#_Toc7637865)

[Figuur 11.1: Schema van een PID regelaar 33](#_Toc7637866)

Code fragmenten

[Code 6.1: PIP commando voor het installeren van OpenCV 15](#_Toc7637832)

[Code 6.2: PIP commando voor het installeren van OpenCV 4.0.0 15](#_Toc7637833)

[Code 6.3: Importeren van de bibliotheken 17](#_Toc7637834)

[Code 6.4: Importeren van de bibliotheken voor de Raspberry Pi camera 17](#_Toc7637835)

[Code 6.5: Code voor het gebruik van een USB camera 17](#_Toc7637836)

[Code 6.6: While lus voor het kunnen sluiten van het programma 18](#_Toc7637837)

[Code 6.7: Code voor het stoppen van OpenCV met een toets 19](#_Toc7637838)

[Code 6.8: Code voor het gebruik van de Raspberry Pi Camera module 19](#_Toc7637839)

[Code 6.9: Code fragment kleur omzetten naar grijstinten 22](#_Toc7637840)

[Code 6.10: Code fragment Gaussian blur en kernel 22](#_Toc7637841)

[Code 6.11: Code fragment threshold 23](#_Toc7637842)

[Code 6.12: Canny edge detection in OpenCV 25](#_Toc7637843)

[Code 6.13: Contouren detecteren en isoleren in OpenCV 26](#_Toc7637844)

[Code 6.14: For-lus om over de contouren te lopen en te filteren 26](#_Toc7637845)

[Code 6.15: Hoekpunten differentiëren in de contouren 27](#_Toc7637846)

Tabellen

[Tabel 6.1: Minimum vereisten voor het vision systeem 14](#_Toc7637869)

Literatuurlijst

Heinisuo, O.-P. (2019, Januari 9). *opencv-python 4.0.0.21*. Opgeroepen op April 25, 2019, van PyPi: https://pypi.org/project/opencv-python/4.0.0.21/

Hildreth, E. C. (1985, September). Edge Detection. *A.I. Memo*(858). Opgeroepen op April 24, 2019, van http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.5715&rep=rep1&type=pdf

NumPy Developers. (2019, April 22). *numpy 1.16.3*. Opgeroepen op April 25, 2019, van PyPi: https://pypi.org/project/numpy/

OpenCV team. (sd). *OpenCV - about*. Opgeroepen op April 25, 2019, van OpenCV: https://opencv.org/about/

Rosebrock, A. (2018, December 3). *imutils 0.5.2*. Opgeroepen op April 25, 2019, van PyPi: https://pypi.org/project/imutils/

The Python Software Foundation. (sd). *Python - about*. Opgeroepen op April 25, 2019, van Python: https://www.python.org/about/

Index

**No index entries found.**

Bijlagen

1. Het instellen van de resolutie vereist wat finesse; de Raspberry Pi camera module ondersteunt slechts een beperkt aantal resoluties. Deze resoluties zijn ook niet beschikbaar bij elke framerate. De USB-webcam werkte het beste bij een resolutie verhouding van 4:3. Ook hier waren de resolutie beperkt, een resolutie van 1440 op 1080 leek het beste te werken. [↑](#footnote-ref-1)