Wet-Cat

Minor project EVD

# Summary

This document will give an explanation of the project Wet-Cat. The project has been made for the minor embedded vision design in the first 8 weeks of the minor. The goal of the project is to use a gun turret equipped with a laser to scare cats away from a yard. The cats will be recognized by a camera using openCV on a laptop.

# Voorwoord

Dit project is gemaakt voor de minor embedded vision design. Het project is een combinatie van vision en embedded programmeren. Het herkennen van objecten gebeurd met openCV terwijl de gun turret met een Arduino wordt aangestuurd. In dit document worden alle aspecten van het project aan de orde gebracht om een duidelijk beeld te geven van wat er tijdens de ontwikkeling van het project gedaan is. Het project is gemaakt op github, de keuze voor dit opensource versie beheer systeemis gemaakt omdat het project voor school is en dus publiek kan zijn, en omdat het een makkelijk systeem is waar documenten in markdown gemaakt kunnen worden en waar de leraar makkelijk bij alle documentatie kan.

# Inhoudsopgave

Test

# Inleiding

In dit document wordt uitleg gegeven over het project “Wet-Cat” dat gemaakt is voor de minor embedded vision design. Het project is gemaakt als oplossing voor het probleem met de hoeveelheid katten binnen de tuin die daar niet horen, maar het project is aangepast voor embedded vision design zodat deze in de klas gedemonstreerd kan worden.

# Functioneel ontwerp

Het functioneel ontwerp is gemaakt op github in markdown, in bijlage A staat het document in markdown, dit document is online beter te zien.

# Technisch ontwerp

Het technisch ontwerp is gemaakt op github in markdown, in bijlage B staat het document in markdown, dit document is online beter te zien [link].

# Vision Ontwerp

Het vision ontwerp is gemaakt op github in markdown, in bijlage C staat het document in markdown, dit document is online beter te zien [link].

# Realisatie

Het realisatie document is gemaakt op github in markdown, in bijlage D staat het document in markdown, dit document is online beter te zien [link].

# Onderzoek

## OpenCV

In het begin van het project is er onderzoek gedaan naar hoe OpenCV in combinatie met QTCreator. Er is gekeken naar hoe OpenCV gebruiken kan worden met QTCreator en wat er allemaal gebeuren moet om OpenCV werkend te krijgen in combinatie met QTCreator. Dit onderzoek is gedocumenteerd in wiki vorm op github [link].

# Testen en resultaten

Tijdens de ontwikkeling van het project is er het een en ander getest. Een van de belangrijkste testen die we hebben gedaan is het calibreren.

## Calibreren

Het calibreren van de gun turret was een uitdagende test, dit komt omdat de gun turret bestuurd word door twee servo motoren, deze motoren zorgen dat de laser een “resolutie” krijgt van 180 bij 180 graden. Op basis van de afstand van de laser tot de muur geeft dit een soort frame waarin de laser kan bewegen.

## Probleem

Het probleem dat opgelost moest worden is dat de de camera, op bijv. 2 meter afstand van de muur, een frame van bijvoorbeeld 1 bij 2 meter, de camera ziet dit als 360 bij 240 pixels. De gun turret heeft op deze afstand een veel groter bereik en moet worden gecalibreerd zodat de pixels van de camera gebruikt kunnen worden om de laser te besturen.

## Oplossing

De oplossing voor dit probleem was om vier punten op de camera als eik punten te gebruiken. Door de laser in beeld te richten op vier verschillende punten en bij ieder punt de camera positie van dat punt door te geven aan de gun turret. De gun turret kan deze dan mappen aan de positie van de servo motoren op dat moment:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Positie | Servo Positie X | Servo Positie Y | Camera Positie X | Camera Positie Y |
| 1 | 50 | 20 | 60 | 60 |
| 2 | 85 | 20 | 300 | 60 |
| 3 | 85 | 40 | 300 | 200 |
| 4 | 50 | 40 | 60 | 200 |

Nadat deze gegevens bekend zijn bij de turret kan deze zichzelf calibreren, dit gebeurd met de volgende berekening:

SX1 = 50 (meest linker positie van de servo)  
SX2 = 85 (meest rechter positie van de servo)  
X1 = 60 (meest linker positie van de camera)  
X2 = 300 (meest rechter positie van de camera)  
Factor = (X2 – X1) – (SX2 – SX1)  
Constante = (Factor \* SX2) – X2

Nadat de Factor en de Contante waardes bekend zijn kan ieder punt worden vertaald met:

X(servo) = X(camera) \* factor + Constante

Dit geld ook voor alle Y waardes.

# Conclusie en aanbeveling

Uiteindelijk is het project goed verlopen. Er zijn wat problemen geweest met het maken van de software en de communicatie tussen de applicatie en de gun turret was niet makkelijk.

Wat erg goed gegaan is, is de structuur van de software en het maken van de gun turret dat in het fablab gedaan is.

De gun turret is gemaakt met oog op de toekomst en we bevelden dan ook aan om deze te gebruiken om in de toekomst een project mee te doen.

# Bijlage