Tampereen ammattikorkeakoulu

Myymälän aktiivinen varastosaldojen seuranta

Markus Virtanen

OPINNÄYTETYÖ
Toukokuu 2019

Tieto- ja viestintätekniikan koulutus
Tietoliikennetekniikka ja tietoverkot

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutus
Tietoliikennetekniikka ja tietoverkot

VIRTANEN MARKUS:
Myymälän aktiivinen varastosaldojen seuranta

Opinnäytetyö 26 sivua, joista liitteitä 1 sivu
Toukokuu 2019

Työn tarkoituksena on tutkia tietokonenäköä tekniikkana ja pohtia, miten tekniikkaa voisi tehokkaammin soveltaa nykyaikaisessa myymäläympäristössä. Työn aihe on opiskelijan itse kehittämä ja pohjautuu itse tehtyihin havaintoihin myymälätyöskentelystä usean vuoden kokemuksen aikana. Työ on toteutettu kevään 2019 aikana.

Projektissa tutustutaan lähtötilanteeseen käymällä läpi saldoissa esiintyviä virheitä ja etsitään niille syitä. Ongelmakohtia lähdetään ratkaisemaan suunnittelemalla laite, joka parantaa saldojen tarkkuutta ja helpottaa niiden seuraamista reaaliaikaisesti. Tavoitteena työssä on tuottaa laite, joka voitaisiin viedä olemassa olevaan myymälään laskemaan annetun alueen tuotteiden lukumäärää.

Työssä käydään läpi myös teoriaa tietokonenäöstä ja erilaisista tekniikoista, joita työssä käytetään hyväksi. Oleellisena osana on myös itse ohjelman kirjoittaminen ja läpikäynti. Myös rakennetun laitteen suorituskykyä arvioidaan työn aikana.

Tulokset työssä ovat lupaavia, sillä yksinkertaisellakin muodontunnistusohjelmalla saadaan jo oikeita mittaustuloksia. Tämän tekniikan huonot puolet käsitellään kuitenkin huolella läpi. Opetusdatan ajaminen matalatehoisella laitteella aiheutti kuitenkin suurempia haasteita. Tähänkin esitellään vaihtoehtoisia menetelmiä työn loppupuolella.

Asiasanat: tietokonenäkö, saldo, suorituskyky, koneoppiminen

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Telecommunications and Networks

VIRTANEN MARKUS:
Actively measuring the inventory of a store

Bachelor's thesis 26 pages, appendices 1 page
May 2019

The purpose of this thesis is to look into computer vision as a technology and find out ways to implement it to a modern store environment. The subject of this project is selected by the writer himself. Many of the observations and conclusions made in the work are based on the students many years of retail experience. The project was carried out in the spring of 2019.

The project covers the current position of technology and inventory management in a typical store and will try to uncover the reasons behind errors in stock numbers. One of the goals of this project is to find a way to make it easier to track the number of items in a specified area. This is done by creating a device that uses computer vision to count items.

The thesis also covers some theory behind the software and technologies that are used. One substantial part of the work is the programming of the device. Performance of the device is also evaluated during the work.

Results of the project are promising. Even with quite simple shape recognition programming it was possible to get correct results during the software's testing. However, the flaws of using only shape recognition are addressed thoroughly. The attempts at running heavier tasks such as training a machine software lead to difficulties with the device's performance capabilities. Solutions to this are also covered in the thesis.

Key words: computer vision, inventory, performance, machine learning

SISÄLLYS

LYHENTEET JA TERMIT

Koneoppiminen	Tekoälyn osa-alue, pyrkii oppimaan ja kehittymään sille syötetystä datasta ilman erillistä ohjelmointia
Hahmontunnistus	Koneoppimisen osa-alue, tunnistaa annetusta kuvasta mallin mukaisia osia tai kaavoja
Tietokonenäkö	Teknologia, joka kehittää tapoja, joilla tietokoneet voivat nähdä ja ymmärtää kuvia ja videoita.
Saldo	Varastossa tai hyllyillä olevien tuotteiden lukumäärä kaupassa.
VNC-yhteys	Etäyhteysohjemisto, jolla saadaan verkon yli hallittua laitteita.
OpenCV	Avoimen lähdekoodin tietokonenäkö- ja koneoppimiskirjasto.

1 JOHDANTO

Lähivuosina niinkin arkiseen asiaan, kuin kaupassa käymiseen on alettu sulauttaa tekniikkaa enemmän ja enemmän. Vielä 2000-luvun alussa olisi ollut käsittämätön ajatus, tarkistaa puhelinapplikaatiosta tai kaupan verkkosivuilta koko myymälän valikoima, tai esimerkiksi tietyn tuotteen kappalemäärä. Tietotekniikan räjähdysmäinen kasvu on tuonut kuitenkin kaiken tämän informaation tavallisen kuluttajan saataville.

Monet työn huomiosta pohjautuvat opiskelijan omaan kokemukseen kaupan alalta.

Tässä opinnäytetyössä tutkitaan mahdollista seuraavaa askelta myymälöiden automaatiossa. Tarkoituksena on toteuttaa laite, joka seuraa määritellyn kokoisen alueen kohdalla olevien esineiden määrää. Vaikka opinnäytetyössä käsitellään myymälää kokonaisuudessaan, ei tarkoituksena ole suunnitella valmista ratkaisua isolle myymälälle.

Opinnäytetyössä käsitellään alkuun tavallisen myymälän nykytilannetta tekniikan ja automaation perspektiivistä. Esitellään myös nykyisiä ratkaisuja ja pohditaan niiden tuottamia haasteita. Opinnäytetyön teoriaosuudessa käsitellään tekniikoita, joita projektissa toteutettava laite hyödyntää. Näihin tekniikoihin kuuluu muun muassa koneoppimista ja tietokonenäön hyödyntämistä. Suunnitteluosuudessa käsitellään ja perustellaan, millaiseen ratkaisuun päädyttiin laitteen toteuttamiseksi. Loppu työ käsittelee projektin toteutusta ja pohdintaa.

2 LÄHTÖTILANNE

Motivaationa tälle projektille on opinnäytetyön kirjoittajan oma kokemus myymälätyöskentelystä ja sen helpottamisesta. Ajatuksena on tutkia tapaa, jolla saataisiin automatisoitua saldojen seurantaa kaupassa tietokonenäköä ja hahmontunnistusta hyväksikäyttäen. Työn tarkoituksena ei ole luoda ihmisiä korvaavaa järjestelmää myymälään, vaan työtä helpottavaa dataa keräävä laite.

2.1 Salojen seuranta päivittäistavarakaupoissa

Nykyisin päivittäistavarakaupan saldojen seuranta pohjautuu täysin tilattujen tuotteiden määrää verrattuna myytyihin tuotteisiin. Tietokannassa oleva saldo muuttuu siis ainoastaan kahdessa pisteessä, kun tavaraa tulee myymälään ja kun se kassalla myydään asiakkaalle.

KAAVIO 1. Tilanteet joissa tuotteiden määrä muuttuu järjestelmässä

Huomionarvoista päivittäistavarakauppojen kohdalla on se, että tieto tuotteiden on usein virheellistä ja aiheuttaa väärän kokoisia tilauksia. Välillä saldot saattavat hävikin tai varkauksien johdosta olla niin virheelliset, että tilausjärjestelmä saattaa tilata tuotteita tuplamäärän tai jättää tilauksen kokonaan tekemättä.

Koska virheelliset hyllysaldot ovat huomionarvoinen ongelma, kuluu tähän henkilökunnan työaikaa päivittäin. Myös asiakastyytyväisyys kärsii, jos tuotteita ei kaupassa ole. Päivittäistavarakauppojen kohdalla tarkkaa tuotteiden määrää ei myöskään ilmoiteta asiakkaalle verkkosivuilla tai applikaatioissa.

2.2 Saldojen seuranta muualla

Myymälät, jotka myyvät esimerkiksi viihde-elektroniikan tai kodinkoneita, usein ilmoittavat saldojen tarkat määrät verkkosivuillansa. Tämä tuottaa asiakkaalle suurta lisäarvoa ja estää turhat matkat, jos tuote onkin päässyt myymälästä loppumaan. Näissäkin yhteyksissä saldojen mittaus on silti manuaalista ja saldojen mittauspisteitä on silti ainoastaan toimitettujen tuotteiden määrä verrattuna myytyihin tuotteisiin.

3 TYÖN TAVOITTEET

Tässä kappaleessa käsitellään lyhyesti työn eri tavoitteet. Koska aihe on laaja ja haastava, asetetaan työlle useampi realistiseksi koettava tavoite. Käsitellään tavoitteiden toteutuvuus viimeisessä kappaleessa kohta kohdalta ja analysoidaan tuloksia tarkemmin.

3.1 Laitteen suunnittelu ja rakennus

Rakennetaan laite, jota voidaan hallita etänä sisäverkosta, ja joka pystyy ottamaan kuvia. Laitteen tulee kyetä käsittelemään kuvia, ja tulkitsemaan niistä hyödyllistä informaatiota. Laiteen koon tulee olla sellainen, että sen saa asennettua tavallisen kaupan hyllyyn kiinni.

3.2 Muotojen tunnistus

Tavoitteena saada laite tunnistamaan jokin tavallinen geometrinen muoto ja saada laite laskemaan näiden muotojen määrä ottamastaan kuvasta. Valitaan muodoksi ympyrä, koska tällöin pystytään testaamaan esimerkiksi tölkkien yläosalla laitteen suoriutumista.

3.3 Hahmon tunnistus opetusdatasta

Pyritään opettamaan laitteelle opetusdatalla tietty esine. Esineeksi valitaan tässäkin tölkin yläosa. Tähän tavoitteeseen lähdetään varauksella, sillä opetusdatan kerääminen ja käsittely saattaa viedä paljon aikaa.

3.4 Laitteen suorituskyvyn analysointi

Tavoitteena seurata, mihin tämän mallinen yhden laitteen ratkaisu riittää ja taipuu. Mitataan suorituskyvyn riittävyyttä ja toteutuksen mahdollisia heikkoja kohtia.

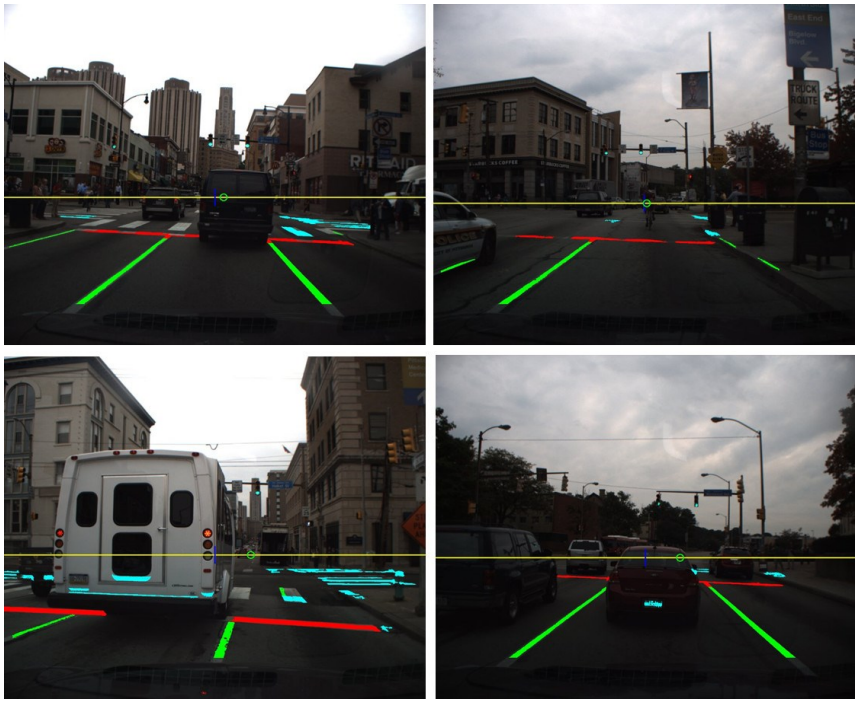
4 TEORIAOSUUS

Tässä kappaleessa käsitellään työssä käytettyä teoriaa. Ensin perehdytään tietokonenäön ja koneoppimisen peruskäsitteisiin. Jälkimmäisissä kappaleissa käsitellään tarkemmin hahmontunnistukseen käytettävää teoriaa.

4.1 Tietokonenäkö

Projektin kannalta oleellisin osa on sensori, jolla saadaan mitattua esineiden määrää halutulla alueella. Sensorina toimii kamera, jonka tuottamaa kuvaa analysoidaan tietokonenäön avulla.

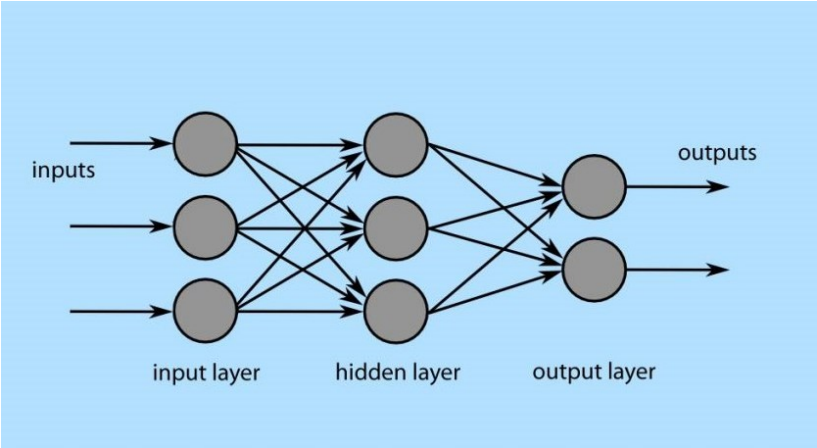
Tietokonenäöllä tarkoitetaan tekniikkaa, jolla tietokoneet saadaan erottamaan informaatiota digitaalisista kuvista tai videosta. Käytännön applikaatioita tälle tekniikalle on automaatiossa paljon, sillä monet työt vaativat visuaalista tarkastelua. Hyvänä esimerkkinä tietokonenäön potentiaalista ovat itseajavat autot. Jotta ajaminen onnistuu ilman kuljettajaa, tarvitsee autossa olevan tietokoneen tunnistaa valtava määrä informaatiota, kuten tie, liikennemerkit ja muut tiellä kulkevat autot.



KUVA 1. Autosta kerättyä dataa (Seo & Rajkumar 2014)

4.2 Koneoppiminen

Koneoppimisella tarkoitetaan ohjelmaa tai ohjelmistoa, joka suorittaa tietyn tehtävän ilman kovakoodattuja ohjeita. Näiden sijaan käytetään opetusdataa, jolla on tarkoitus luoda vertailukohta oikealle datalle. Kuvassa X havainnollistetaan neuroverkon toimintaa.



KUVA 2. (Rao 2018)

Tietokonenäön perspektiivistä koneoppiminen on kriittisessä roolissa. Hahmontunnistus olisi käytännössä mahdotonta toteuttaa, jos jokaiselle esineelle pitäisi erikseen ohjelmoida tunnistukseen tarvittavat parametrit. Tämän sijaan ohjelmistolle voidaan antaa valtava määrä opetusdataa, jonka sisältö tiedetään etukäteen.

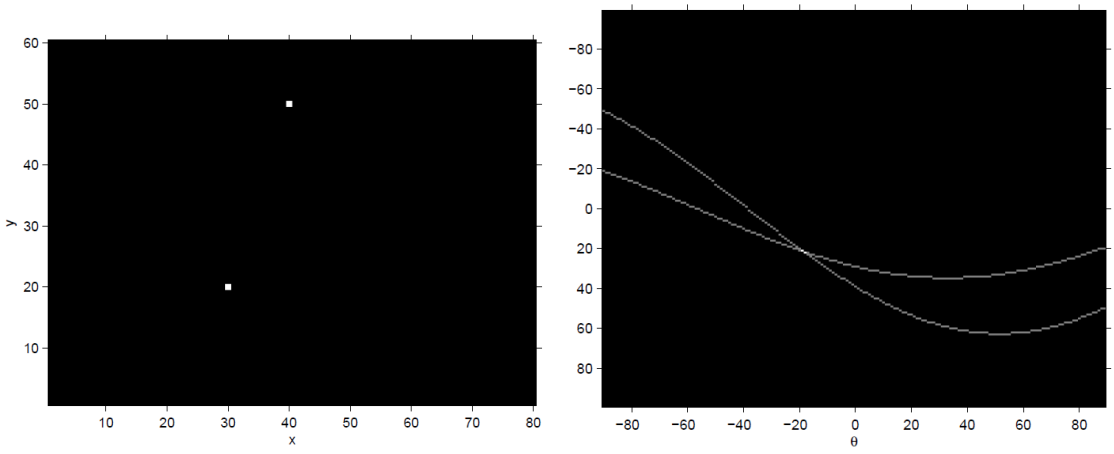
4.3 Reunan tunnistus Hough -muunnoksella

Hough -muunnos etsii yksittäisten reunapisteiden välillä kulkevan suoran muuntamalla pisteet hough -avaruuteen. Muokataan suoran yhtälö, kaavan 1 mutoiseksi. Muunnoksessa käytetään kaavan 2 mukaista suoran yhtälöä.

$$y = x \cdot \cos\theta + y \cdot \sin\theta \quad (1)$$

$$y = -\frac{\cos\theta}{\sin\theta} \cdot x + \frac{r}{sin\,\theta} \quad (2)$$

Muunnoksessa jokaiselle reunapisteelle etsitään jokainen mahdollinen suora, joka voi kulkea pisteen läpi. Kaaviossa 2 esitetään hough -muunnos pisteille p01 ja p02. Vasemmanpuoleisessa kaaviossa on esitettynä pisteiden sijainti koordinaatistossa. Oikealla pisteille on tehty Hough -muunnos.



Kaavio 2. Pisteet p01 ja p02 vasemmalla, oikealla pisteiden Hough -muunnos. (Line Detection by Hough transformation 2009)

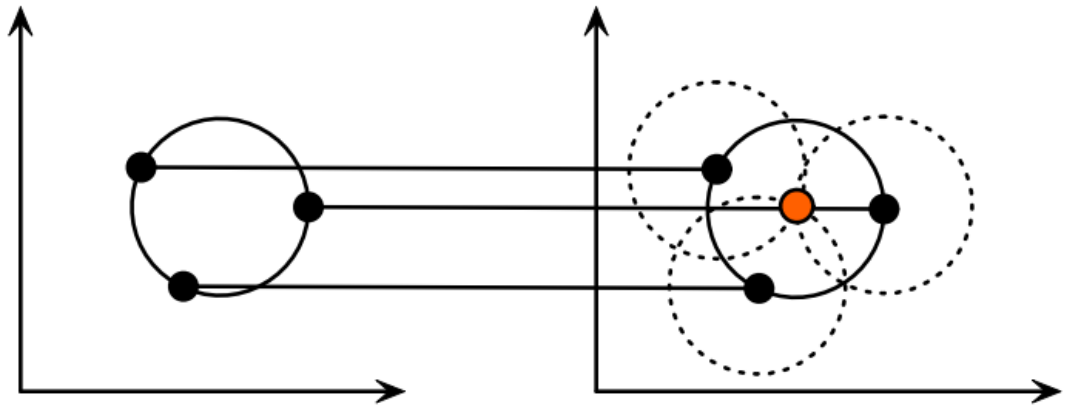
Kaavion 2 oikean puolen kuvaajassa käyrien leikkauskohdan avulla pystytään määrittämään suora, joka kulkee pisteiden läpi. Tässä esimerkissä on vain kaksi pistettä, mutta todellisessa reunakartassa reunapisteitä olisi useita, joille kaikille tehtäisiin muunnos erikseen.

4.4 Ympyrän tunnistus Hough -muunnoksella

Ympyrän tunnistus Hough -muunnoksella noudattaa samoja työvaiheita kuin reunantunnistus. Haasteena kuitenkin on, että tunnistettavan ympyrän säde tulee tietää etukäteen. Ympyrän kaavana käytetään alla olevaa kaavaa (kaava 3),

$$(x - a)^2 + (y - b)^2 = r^2 \tag{3}$$

jossa muuttujat a ja b määrittävät ympyrän keskipisteen koordinaatit. Reunakartalle piirtyneet reunapisteet viedään koordinaatistoon, ja jokaiseen reunapisteeseen piirretään tiedetyn säteen (r) avulla ympyrä.



Kaavio 3. Ympyrän tunnistaminen kaaren reunapisteillä. (Circle Hough Transform (CHT). N.d.)

Kaaviossa 3 esitetään reunapisteiden avulla ympyrän keskipisteen määrittäminen tiedetyn säteen avulla. Mikäli sädettä ei tiedetä, joudutaan toimenpide suorittamaan useaan kertaan eri säteen arvoilla. Tällöin saadaan samasta kuvasta tunnistettua monen kokoisia ympyröitä.

5 SUUNNITTELU

Tarkoituksena on rakentaa pieni laite, jolla pystytään tutkimaan ja testaamaan tietokonenäköä myymäläympäristössä. Koko myymälän kattavaa konenäköjärjestelmää varten tarvittaisiin valtava määrä prosessointikykyä ja laitteita, joten keskitytään tässä opinnäytetyössä luomaan yhden kameran sisältävä laite, jolla voidaan seurata yhden tuotteen lukumäärää annetulla hyllypaikalla.

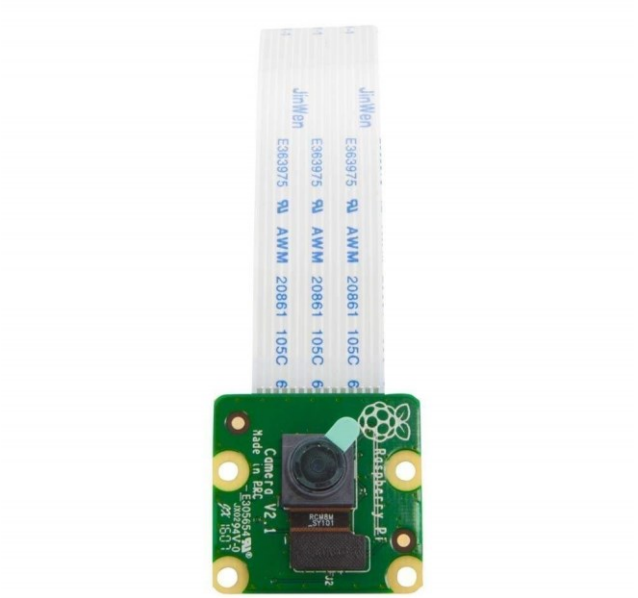
Laite, jolla työ toteutetaan, on Raspberry Pi 3 model B+ (Kuva 3). Kyseessä on pieni, yhden piirilevyn tietokone, joka on tarkoitettu ohjelmoinnin opetteluun ja pieniin projekteihin.

KUVA 3. Raspberry Pi 3 model B+

Tällaisenaan laiteella pystyisi käsittelemään kuvia, jotka on otettu toisaalla ja sitten siirretty laitteelle esimerkiksi verkon yli. Kuitenkin tarkoituksena on luoda yksi laite, jolla hoidetaan koko prosessi, joten laitteelle pitää asentaa kamera.

5.1 Kamera

Jotta laite saa otettua kuvia, tarvitaan laitteelle kamera. Vaihtoehtoina kamera-laitteelle oli, joko käyttää mitä tahansa USB-kameraa tai ostaa laitteelle virallinen lisäosa Raspberry Pi Camera Module (Kuva 4). Päädyttiin investoimaan viralliseen lisäosaan, sillä tarpeeksi tarkkaa USB-liittimellä varustettua kameraa ei valmiiksi löytynyt. Arveltiin myös, että ohjelmistotuki viralliselle lisäosalle on todennäköisesti kattavampi.



KUVA 4. Raspberry Pi Camera Module

Huomioitavaa on myös se, että virallisen kameramoduulin johto on paljon lyhempi kuin kolmannen osapuolen web-kamera tyyppisissä ratkaisuissa. Tämä tarkoittaa sitä, että käytettäessä virallista kameramoduulia, on koko paketti hyvin tiivis. Tämä taas johtaa siihen, että koko tietokone kameroineen tulee kiinnittää tarkkailtavan alueen yläpuolelle.

5.2 Verkkoyhteys

Laitteelle tarvitaan myös verkkoyhteys, jotta mitattu data saadaan luettua. Raspberry Pi:n saa kytkettyä verkkoon, joko langattomasti tai kaapelilla. Työssä käytetään langatonta verkkoyhteyttä, jonka kautta muodostetaan laitteelle VNC-etäyhteys. VNC:llä saadaan Raspberryn työpöytä näkymään pöytätyöasemalla.

5.3 Käyttöjärjestelmä

Käyttöjärjestelmänä Raspberry Pi tietokoneelle käytetään Raspbian Stretch with desktop käyttöjärjestelmää. Tämä sopii hyvin testaamiseen ja toteutukseen, sillä käyttöjärjestelmästä löytyy työpöytänäkymä. Tämä nopeuttaa testaamista, koska laitteella itsellään pystytään esikatselemaan kerättyä dataa. Kyseisestä käyttöjärjestelmästä löytyy myös valmiiksi tuki kameralle, joka asennetaan laitteeseen.

5.4 OpenCV

Kun laite on muuten valmis käytettäväksi, on tarkoitus asentaa avoimen lähdekoodin tietokonenäkökirjasto OpenCV. Kirjasto pitää sisällään 2500 koneoppimis ja tietokonenäölle suunniteltua algoritmia. Kyseistä kirjastoa käytetään suurissa yrityksissä kuten Google, Yahoo ja Microsoft. OpenCV:n käyttöön päädyttiin kirjaston yhteensopivuuden raspberry -laitteen kanssa. (OpenCV 2019)

6 TOTEUTUS

Tässä kappaleessa käsitellään projektin toteutusvaihetta. Aloitetaan käsittely laitteen ja testiympäristön rakentamisella. Myöhemmissä kappaleissa käydään läpi Open CV:n asentaminen ja käytetään tätä tunnistamaan kuvaa.

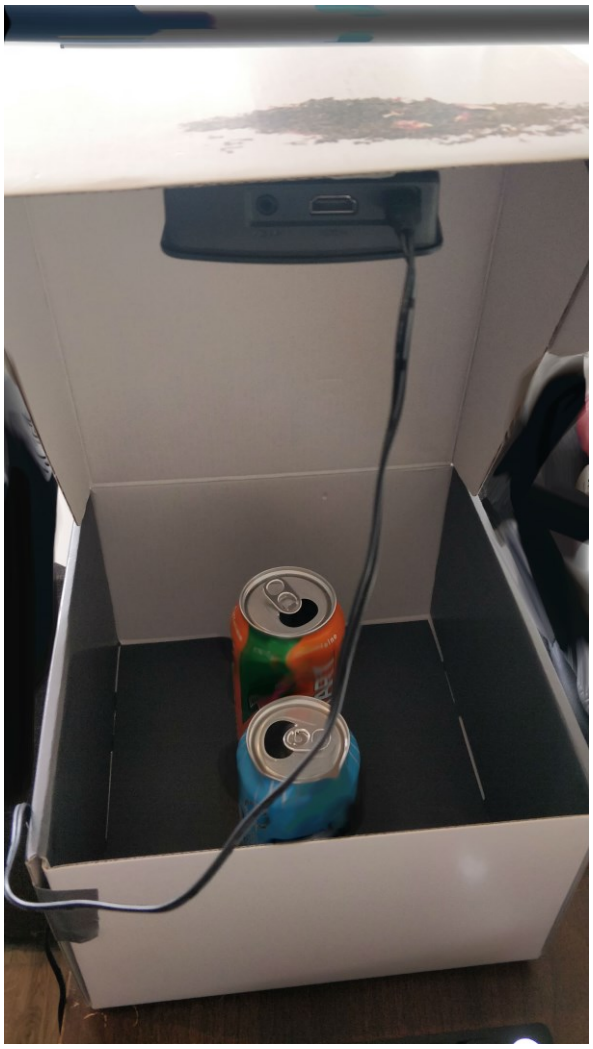
6.1 Laitteen rakentaminen

Laitteen osien asentaminen oli hyvin yksinkertaista, ainoa erikseen kytkettävä osa oli kamera. Laitteelle päätettiin kuitenkin hankkia vielä suojakotelo, sillä kameraa ei muuten olisi saanut turvallisesti kiinnitettyä. Kuvassa 5 näkyy laite kaikkine osineen käyttövalmiina.

KUVA 5. Raspberry Pi lisälaitteineen käyttövalmiina

6.2 Testiympäristö

Laitteen testaamista varten rakennettiin yksinkertainen alusta, johon laite saatiin kiinnitettyä kamera alaspäin. Kamera osoittaa kohti reunojen rajaamaa tumman väristä aluetta, johon sijoitetaan laskettavat esineet. Kuvassa 6 mittalaite on kiinnitetty testialustan yläosaan ja mittausalustalle on asetettu 2 tölkkiä.



KUVA 6. Testialusta

Mittausalustan väriksi valittiin mahdollisimman tumma väri, jotta tölkkien reunat tulisivat kunnolla esiin kuvassa. Tässä vaiheessa huomattiin työssä käytettävä kamera ei soveltuisi kovin hyvin myymälän olemassa oleviin hyllyihin, sillä se piti asettaa paljon korkeammalle kuin alun perin arvioitiin. Tämä voitaisiin käytännön toteutuksessa korjata vaihtamalla kameraan, jossa on laajakulmaisempi objektiivi.

6.3 OpenCV asennus & käyttöönotto

OpenCV:n asennus on pitkä ja monivaiheinen prosessi. Verkkosivuilla www.pyimagesearch.com löytyi valmiiksi ohjeet, kuinka asennus onnistuu Raspbian Stretch –käyttöjärjestelmälle (Rosebrock 2017). Näitä ohjeita seuraamalla saatiiin OpenCV asennettua, aikaa kului kuitenkin useampi tunti laitteen matalan suorituskyvyn vuoksi. Työssä asennettava versio OpenCV:stä on OpenCV 3 ja asennus suoritettiin Python 3 -kielelle.

6.4 Muodon tunnistus

Kirjoitettiin pythonilla ohjelma, joka ottaa kuvan ja etsii kuvasta ympyrän muotoisia esineitä. Kuvassa 7 näkyy ohjelman osa, joka ottaa kuvan PiCamera-funktiota käyttäen ja tämän jälkeen tallentaa sen.

```
camera = PiCamera()
camera.start_preview()
sleep(2)
camera.capture('/home/pi/circledetection/testimage.jpg')
```

KUVA 7. Ohjelman alku, joka tallentaa kuvan

Kun kuva on tallennettu, ladataan se img-nimiseen muuttujaan. Seuraavaksi kuvaa käsitellään OpenCV:n työkaluilla (kuva 8). Tämän jälkeen kuvasta poistetaan terävät reunat sumentamalla kuvaa medianBlur-toiminnolla. Lopuksi kuva muunnetaan vielä mustavalkoiseksi.

```
img = cv2.imread('testimage.jpg',0)
img = cv2.medianBlur(img,5)
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)
```

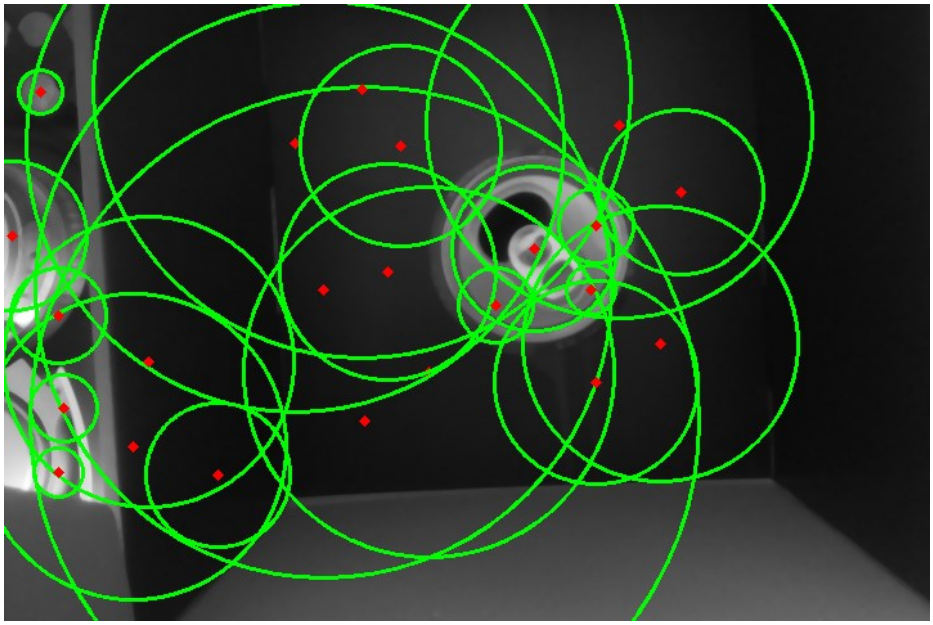
KUVA 8. Kuvan muokkaus

Seuraavaksi suoritetaan itse muodon tunnistaminen. Tähän käytetään HoughCircles-funktiota (kuva 9).

```
circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,50,
                           param1=50,param2=30,minRadius=50,maxRadius=100)
```

KUVA 9. ympyröiden tunnistaminen kuvasta

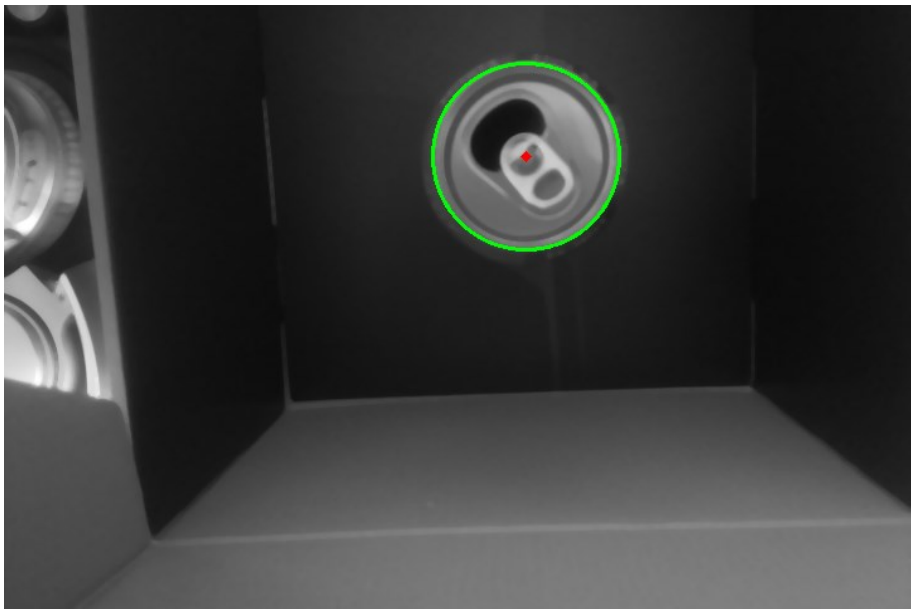
Loppu ohjelma tulostaa ympyröiden lukumäärän ja piirtää kuvaan ympyrälle keskipisteen ja kehän. Kuvassa 9 esiintyvässä funktiossa on monta määritettävää parametria. Kiinnitetään erityistä huomiota min- ja maxRadius-parametreihin. Kuten aiemmin kappaleessa 4.4 käsiteltiin, Houg -muunnos tarvitsee etukäteen etsittävän ympyrän säteen. Selvitetään tölkin kannelle säde ottamalla testikuva (kuva 10).



KUVA 10. Testikuva

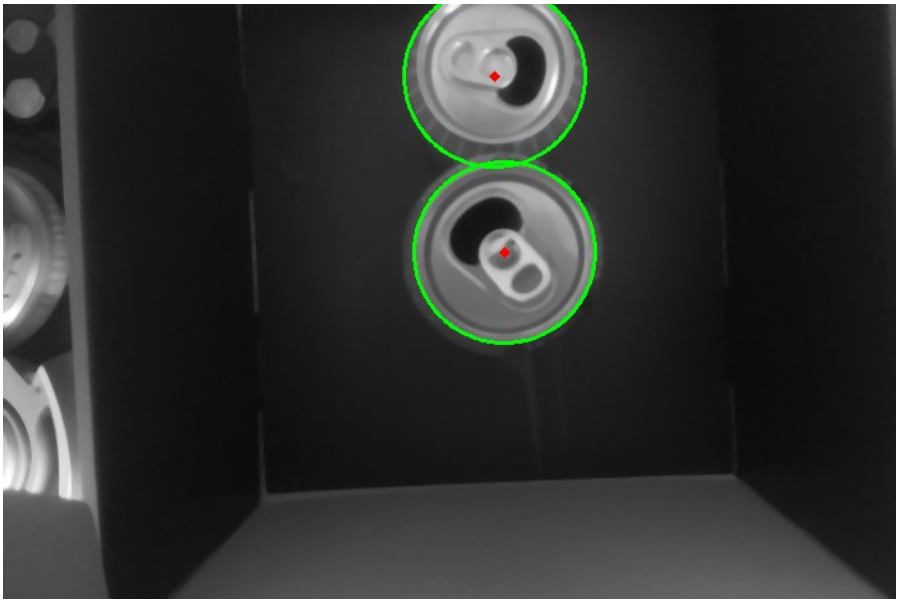
Kuvasta 10 nähdään, että ympyrän tunnistus on todella epätarkkaa, jos ympyrän säteelle ei määrittele reunaehtoja. Kuvasta nähdään, että myös tölkin kansi on tunnistettu. Laskettiin pikselimäärä ympyrän keskustasta ympyrän kehälle ja tulokseksi saatiin noin 75 pikseliä. Koska tämä arvo on kuvasta tarkistettu ja summittainen lisätään minimi- ja maksimiarvoon 25 pikselin virhe. Saadaan kuvassa 9 näkyvä minRadius=50 ja maxRadius=100.

Kun etsittävälle ympyrälle on annettu tarkemmat reunaehdot, saadaan huomattavasti parempia tuloksia. Kuvassa 11 nähdään onnistunut tölkin kannen tunnistus. Kuvasta huomataan myös, että kameran kohdistus on hieman pielessä, kuvan vasemmalta puolelta näkyy muutakin kuin reunaa.



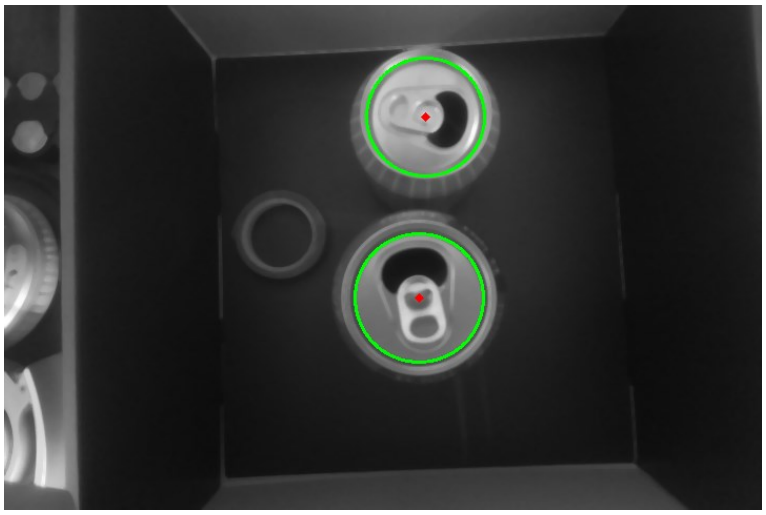
KUVA 11. Onnistunut ympyrän tunnistus

Kun yksi tölkki on tunnistettu onnistuneesti, testataan vielä useampaa tölkkiä vierekkäin. Kuvasta 12 nähdään, että myös useamman tölkin tunnistaminen onnistuu laitteelta. Vaikka ylempi tölkki on hieman ulkona kuvasta, onnistuu ohjelma löytämään riittävästi reunapisteitä.



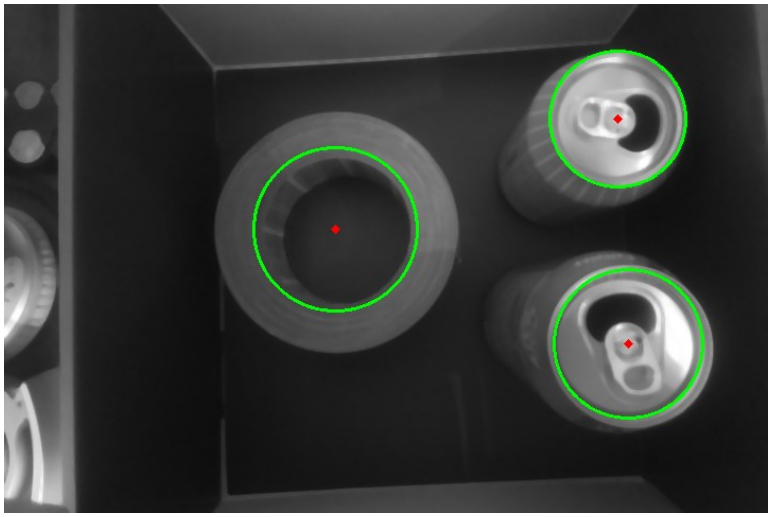
KUVA 12. Kahden tölkin tunnistus.

Jos järjestelmä luotaisiin pelkästään muodon tunnistamisen ympärille, tuotettava data ei olisi kovin luotettavaa. Demonstroidaan tätä muutamalla satunnaisella ympyrän muotoisella esineellä ja asetetaan ne vaikeuttamaan mittausta (kuva 13).



KUVA 13. Pieni teippirulla mittausta vaikeuttamassa

Kuten kuvasta 13 huomataan, pieni teippirulla ei häirinnyt mittausta. Tämä johtuu kuitenkin vain siitä, että teippirullan säde on määritettyjen arvon ulkopuolella. Vaihdetaan suurempaan teippirullaan ja toistetaan koe (kuva 14).



KUVA 14. Suuri teippirulla vaikeuttamassa mittausta

Kuvasta 14 huomataan että suuremman teippirullan sisäkehä sattuu olemaan juuri sopivan kokoinen, jotta ohjelma tunnistaa sen. Tässä tapauksessa saataisiin siis virheellinen tulos.

6.5 Hahmon tunnistus opetusdatasta

Opetusdataa lähdettiin käsittelemään OpenCV:n sisältämillä työkaluilla. Ensin kerättiin 10 eri kuvaa tölkin kannesta. Tämän jälkeen haettiin negatiiveja osoitteesta github.com/sonots/tree/master/data/negatives. Negatiivit tässä tapauksessa tarkoittavat kuvia, jotka eivät sisällä etsittävää hahmoa.

Opetusdatalle onnistuttiin määrittämään koordinaatit, joissa etsittävä hahmo esiintyy. Tämän jälkeen yritettiin ajaa opencv_traincascade-komentoa, Raspberry palautti virheen, josta kävi ilmi välimuistin loppuneen kesken. Todettiin että opetusdatan ajaminen pienellä tietokoneella, ei todennäköisesti tule onnistumaan. Opetusdatasta luotava xml-tiedosto olisi kuulunut ajaa tehokkaammalla laitteella, kuten pöytätietokoneella. Koko ympäristön asentamiseen toiseen kertaan ei kuitenkaan aika riittänyt, joten tämä osuus jäi suurilta osin suorittamatta.

7 POHDINTA

hahmontunnistuksella on selvästi paikka tulevaisuuden kaupan toiminnassa. Tarkkuuden pitää todennäköisesti moninkertaistua nykyiseen, sillä jokaisen tuotteen yläpuolelle asetettavan sensorin asentaminen ei ole realistista. Tällä hetkellä tällaista tekniikkaa voitaisiin hyödyntää kalliiden tuotteiden kohdalla, mutta jokaisen tuotteen laskemiseen tällä menetelmällä ei tarvetta varmasti ole. Jos tämä kaikki toiminnallisuus saataisiin implementoitua esimerkiksi valvontakamerajärjestelmään, saataisiin heti toimivampi ja kustannustehokkaampi ratkaisu.

7.1 Laitteen suorituskyvyn arviointi

Pelkästään muodontunnistukseen riittää jopa pienen tietokoneen suorituskyky. Pientä viivettä huomataan kuvan ottamisessa ja ohjelman ajossa, mutta tulokset saatiin ohjelmasta ulos joka kerralla. Kuitenkin jos tällainen muotoja tunnistava järjestelmä rakennettaisiin, olisi se varmasti kustannustehokkaampaa toteuttaa yhdellä tehokkaalla prosessointiyksiköllä, johon kytkettäisiin kaikki järjestelmään kuuluvat kamerat.

Todellinen potentiaali tälle tekniikalle on kuitenkin koneoppimisen tuottamassa informaatiossa ja sen mahdollistamassa hahmontunnistuksessa. Vaikka tätä puolta ei tässä opinnäytetyössä päästy käsittelemään kunnolla, on selvää että tulevaisuuden järjestelmissä tulee olemaan koneoppimisen tuottamaa dataa. Tämän avulla

LÄHTEET

Seo, Y & Rajkumar, R. 2014. A vision system for detection and tracking of stop-lines. Luettu 2.4.2019.
http://www.cs.cmu.edu/~youngwoo/research.html

Rao, V. 2018. Difference Between AI, Machine Learning And Deep Learning. Luettu 2.4.2019.
http://www.ulalalab.com/difference-between-ai-machine-learning-and-deep-learning/

Line Detection by Hough transformation. 2009. Luettu 15.4.2019.
http://web.ipac.catech.edu/staff/fmasci/home/asro_refs/HoughTrans_lines_09.pdf

Circle Hough Transform (CHT). N.d. [web-sivu]. Luettu 15.4.2019.
https://www.cis.rit.edu/class/simg782.old/talkHough/HoughLecCircles.html

OpenCV team. 2019. [web-sivu]. Luettu 3.4.2019.
https://opencv.org/about/

Rosebrock, A. 2017. Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi. Luettu 17.4.2019.
https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

LIITTEET

Liite 1. Ympyrän tunnistukseen käytetty ohjelmakoodi

```
import cv2
import numpy as np
from time import sleep
from picamera import PiCamera

#Capturing the image using picamera

camera = PiCamera()
camera.start_preview()
sleep(2)
camera.capture('/home/pi/circledetection/testimage.jpg')

#Opencv part of the code

img = cv2.imread('testimage.jpg',0)            #Loading the image
img = cv2.medianBlur(img,5)                    #Blurring the image
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)   #Turning the image to grayscale

#Detecting circles using HoughCircles
circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,50,
                           param1=50,param2=30,minRadius=50,maxRadius=100)

circles = np.uint16(np.around(circles))
amount = 0
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)
    amount=amount+1
print("number of detected circles:",amount)
cv2.imshow('detected circles',cimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```