Elias Saarelainen

Konenäkö pellettikuljettimeen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

7.5.2025

Tiivistelmä

Tekijä:	Elias Saarelainen
Otsikko:	Konenäkö pellettikuljettimeen
Sivumäärä:	31 sivua
Aika:	7.5.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Pääaine:	Automaatiotekniikka
Ohjaaja(t):	Lehtori Matti Välikylä

Tässä insinöörityössä kehitettiin konenäköjärjestelmä Myyrmäen Metropolia Ammattikorkeakoulun automaatiolaboratorioon. Pellettikone siirteli puisia pellettejä yläsäiliöstä alasäiliöön liukuhinnalla. Tavoitteena oli, että konenäköjärjestelmä tunnistaisi liukuhihnalla kulkevat pelletit ja mahdolliset väärät esineet.

Konenäköjärjestelmä kehitettiin Raspberry Pi 4-minitietokoneella ja ohjelmistona käytettiin Thonny Python -ohjelmointiympäristöä. Ohjelmaan ladattiin YOLOv5-konenäkömalli ja se opetettiin tunnistamaan pelletit ja muut liukuhihnalla kulkevat esineet. Konenäkömalli opetettiin itse otetuilla valokuvilla esineistä. Opetettaviin esineisiin tehtiin merkintä Label-Studio työkalulla.

Opinnäytetyön kehittämisprosessin lopputuloksena konenäköjärjestelmä tunnisti oikein pellettikoneen liukuhihnalla olevat kappaleet.

Asiasanat: Automaatio, Raspberry Pi 4, Konenäkö, Python

Abstract

Author:	Elias Saarelainen
Title:	Machine vision for pellet conveyor
Number of Pages:	31 pages
Date:	7 May 2025
Degree:	Bachelor of Engineer
Degree Program:	Electrical and automation engineering
Major:	Automation engineering
Instructor(s):	Lecturer Matti Välikylä

This engineering thesis focused on developing a machine vision system for the automation laboratory at Metropolia University of Applied Sciences in Myyrmäki. The pellet machine transported wooden pellets from the upper container to the lower container via a conveyor belt. The goal was for the machine vision system to recognize the pellets moving on the conveyor belt and detect any foreign objects.

The machine vision system was developed using a Raspberry Pi 4 minicomputer, and the Thonny Python programming environment was used for software development. The YOLOv5 machine vision model was installed in Thonny and trained to identify pellets and potential foreign objects. The model was trained using self-captured images, with objects labeled using the Label Studio annotation tool.

As a result, the machine vision system identified the objects on the conveyor belt of the pellet machine.

Keywords: Automation, Raspberry Pi 4, Machine vision, Python

Sisällys

1 Johdanto

Tässä opinnäytetyössä kehitetiin konenäköjärjestelmä Myyrmäen Metropolia Ammattikorkeakoulun automaatiolaboratorion pellettikuljettimen liukuhihnan valvonnan helpottamiseksi. Tässä opinnäytetyössä hyödynnetään konenäköä ja tekoälyä liikkuvien pellettien ja muiden kappaleiden havaitsemiseen ja tunnistamiseen kuljettimen liukuhihnalla.

Opinnäytetyön kehittämisalustana käytettiin Raspberry Pi 4 –minitietokonetta, joka käyttää konenäköohjelmaa Thonny-nimisessä Python-ympäristössä. Konenäköjärjestelmän kamerana käytettiin koulun tarjoamaa Raspberry PI Camera –moduulia.

Raspberry Pi 4 -minitietokoneessa käytettiin konenäkömallina YOLO (You Only Look Once) -neuroverkkoa, joka toimii Label Studio -merkintätyökalun koulutusdatan avulla. YOLO vertailee opetettuja esineitä uuteen videokuvasyötteeseen tunnistamalla siitä samankaltaisia piirteitä, joita se oppi koulutusvaiheessa.

Tämä insinöörityö sisältää teoreettisen ja käytännöllisen osan. Teoreettisessa osassa esitellään konenäön periaatteita, YOLO-neuroverkon prosessia ja automaation hyödyntämistä teollisuudessa. Lisäksi esitellään konenäköjärjestelmässä käytettävät laitteet ja ohjelmistot. Opinnäytetyön soveltavassa osassa raportoidaan konenäköjärjestelmän kehittämisprosessia pellettikuljettimeen.

2 Konenäön kehitys ja sovelluskohteet

2.1 Konenäön Perusteet ja Toimintaperiaatteet automaatiossa

Konenäkö on teknologia, jonka avulla koneet voivat analysoida ja ymmärtää visuaalista informaatiota ympäristöstään. Esimerkiksi pellettikuljettimessa konenäköjärjestelmä tarkkailee liukuhihnalla kulkevia puusta tehtyjä pellettejä ja tunnistaa mahdolliset väärät kappaleet, jotka voisivat häiritä prosessin toimintaa tai aiheuttaa vaurioita laitteistossa. Konenäköjärjestelmän tavoitteena on varmistaa, että liukuhihnalla kulkee vain oikeat kappaleet prosessia varten. [17]

Konenäköjärjestelmä toimii käyttämällä kameraa, sopivaa valaistusta ja tietokoneen kuvankäsittelyohjelmaa. Hyvä valaistus on todella tärkeää, koska konenäköjärjestelmät tarvitsevat riittävästi valoa havaitakseen yksityiskohdat tarkasti ja heikko valaistus voi heikentää videokuvasyötteen kuvanlaatua. Heikko valaistus voi luoda varjoja ja johtaa virheellisiin objektin tulkintoihin, kuten väärin tunnistettuihin esineisiin tai esineiden muotojen ja kokojen virheelliseen arviointiin. Pellettikuljettimessa väärin tunnistetut esineet voisivat vaikuttaa toimintaprosessiin esimerkiksi lajittelemalla väärät esineet vääriin luokkiin. [17]

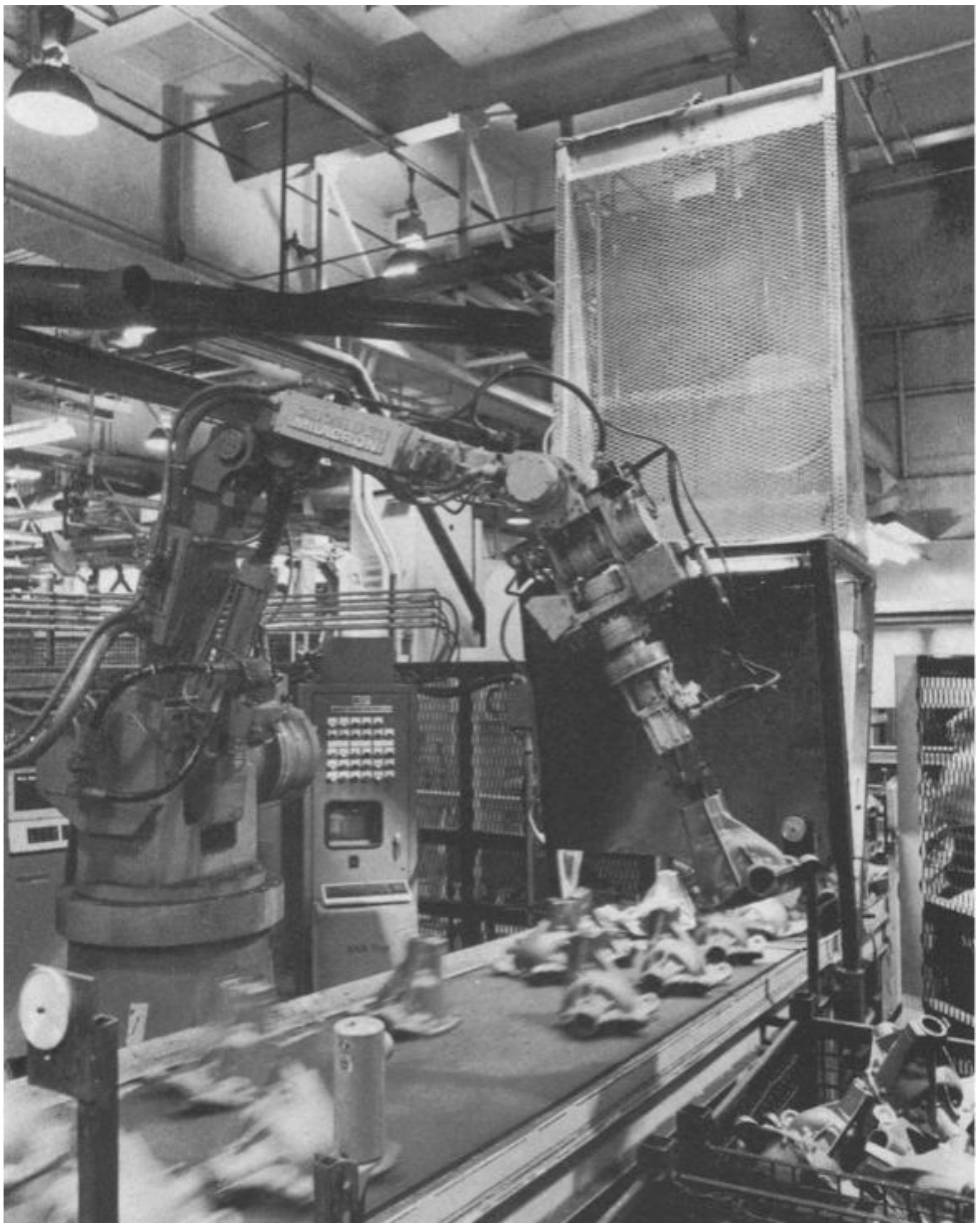
Konenäköjärjestelmän tehokkuus riippuu sen kyvystä käsitellä dataa ympäristöolosuhteissa reaaliaikaisesti. Esimerkiksi, jos liukuhihna liikkuu liian nopeasti, se voi vaikeuttaa kameran kykyä ottaa tarkkoja kuvia esineistä. Laitteiden, joita konenäkö valvoo pitää toimia optimaalisella tavalla, jotta konenäkö toimii oikein ja tarkasti. [17]

2.2 Konenäön historia ja kehityskaari

Konenäön ensimmäiset tutkimukset alkoivat 1960-luvulla, jolloin Larry Roberts julkaisi väitöskirjassaan idean 3D-geometrisen datan tunnistamista 2D-perspektiivikuvista. Larryn tutkimus loi pohjan tuleville konenäköjärjestelmille ja myöhemmin aloitettiin tutkimaan todellisten esineiden käsittelyä. Tutkijat keskittyivät

reunanhavaitsemiseen ja segmentointiin eli kuvan jakamista osiin tai alueisiin. Merkittävä harppaus kehityksessä oli David Marrin kehittämä viitekehys vuonna 1978, jossa hän keksi pohja-ylös-lähestymistavan, jossa ensiksi käsiteltiin 2D-kuvista matalan tason algoritmeilla, kuten reunasegmentoinneilla, ja sitten siirryttiin 3D-mallinnusta kohti. [1]

Autonvalmistaja General Motors oli ensimmäisiä yrityksiä, joka otti konenäköjärjestelmän käyttöön tuotannossaan. Vuonna 1981 yritys kehitti ensimmäisensä konenäköjärjestelmän Ontarion valimossaan ja järjestelmä pystyi lajittelemaan kuutta erilaista metallista kappaletta kuljetinhihnalta. Konenäön avulla robotit pystyivät lajitella 1 400 kappaletta tunnissa (Kuva 1). [11]



Kuva 1. Milacron T3 -robotit käsittelevät metallikappaleita konenäön avulla General Motorsin valimossa Kanadassa vuonna 1981. (Springer, 2023).

Suomessa konenäköä on kehitetty jo 1980-luvusta lähtien Oulun yliopistossa, jossa työskentelee monta asiantuntijaa, joista yli 80 prosenttia on muualta kuin Suomesta. Konenäköä Suomessa käytetään esimerkiksi kauppojen pullonpalautusautomaateissa. Konenäön avulla tunnistetaan pullon tai tölkin muoto ja rahallinen arvo. Palautetut tölkit tai pullot jaetaan automaattisesti oikeisiin kierrätyskategorioihin. [9]

1980-luvulta 2000-luvulle konenäköjärjestelmien kehitys eteni vähitellen tietokoneiden laskentatehon kasvun ansiosta. 2000-luvulla internetistä sai enemmän kuvadataa erilaisista esineistä, mikä helpotti konenäkömallien koulutusta. Myös koneoppimisen algoritmit kehittyivät, ja esimerkiksi kuvioiden tunnistus kehittyi merkittävästi. Konenäköjärjestelmät pystyivät tunnistamaan ja luokittelemaan esineitä luotettavammin kuvadatan ja koneoppimismenetelmien kehityksen ansiosta. Lisäksi kamerateknologia kehittyi 2000-luvun alussa, kun kuvakennot ja optiikat parantuivat huomattavasti ja pystyttiin käyttämään parempaa resoluutiota kuvienkäsittelyyn. [7]

2010-luvulla konenäkö kehittyi merkittävästi, koska syväoppimisen algoritmit yleistyivät. Erityisesti konvoluutioneuroverkot (CNN, Convolutional Neural Networks) mahdollistivat luotettavamman ja nopeamman kuvantunnistuksen. Konvoluutioneuroverkkoja käytettiin esimerkiksi itseajavissa autoissa. [18]

Nykyisin konenäköjärjestelmät ovat kehittyneet monilla automatisoinnin alueilla ja sen sovellusten käyttö on laajentuneet huomattavasti. Merkittävämmät konenäköjärjestelmien alat ovat teollisuus, terveydenhuolto ja autonomiset autot. [7]

2.3 Konenäön hyödyntäminen teollisuusautomaatiossa

Konenäöstä on tullut tärkeä osa teollisuusautomaatiota, sillä se mahdollistaa tarkan laadunvalvonnan ja auttaa tunnistamaan virheitä tuotantoprosesseissa. Erityisesti valmistusteollisuudessa konenäköjärjestelmät ovat tärkeitä, sillä ne auttavat valvomaan, että oikeat kappaleet liikkuvat prosesseissa. Konenäköjärjestelmät auttavat tunnistamaan virheitä kappaleissa, ja virheelliset kappaleet voidaan poistaa prosessista ennen kuin ne aiheuttavat vahinkoa tai ongelmia tuotantoprosessissa. [16]

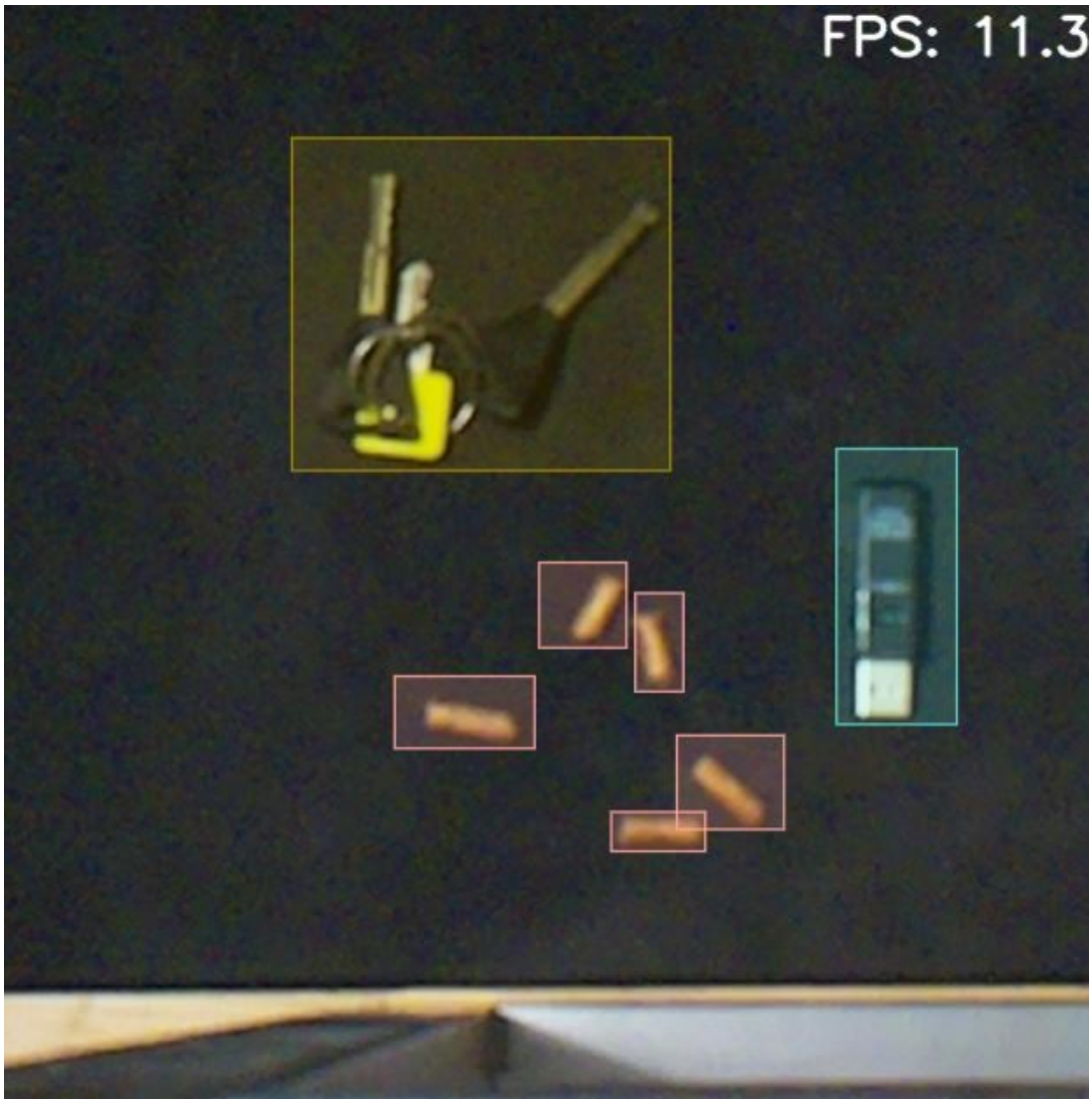
Tuotantoteollisuudessa konenäkö parantaa valvonnan laatua ja tehokkuutta. Konenäköjärjestelmät voivat tarkastella kappaleita ja automaattisesti hylätä vialliset tuotteet. Konenäköjärjestelmä estää virheellisen tuotteen kulkemista asiakkaille ja vähentää tarpeettomia kustannuksia tuotannon valvonnassa. Konenäköautomaation avulla voidaan varmistaa tarkempi laadunvalvonta ja sujuvampi tuotantoprosessi. [16]

Konenäköä voi hyödyntää muillakin aloilla, kuten lääketieteellisissä laitteissa ja logistiikassa. Sitä voidaan käyttää esimerkiksi tuotteiden pakkaamisessa ja lähetysten tarkastelussa, joka tehostaa kuljetusprosessia ja auttaa oikeiden pakettien lähettämistä oikeisiin osoitteisiin. Konenäköjärjestelmät ovat myös tärkeitä itseajavien autojen kehityksessä. Itseajavat autot analysoivat konenäön avulla ympäristönsä ja seuraavat liikennettä. Konenäön avulla auto havaitsee tien, liikennevalot, ihmiset, toiset autot ja mahdolliset esteet. [16]

2.4 Konenäön kouluttaminen ja tunnistusprosessit

Konenäköjärjestelmän käyttö edellyttää koulutusmallia, jonka avulla koneelle opetetaan opetuskuvilla konenäkömalli, jotta se pystyy tunnistamaan ja luokittelemaan esineitä kuvista. Tässä insinöörityössä opetettiin konenäköjärjestelmää tunnistamaan puisia pellettejä ja esineitä, joita ei haluta liikkuvan pellettikoneen liukuhihnalla esim. nappikuulokeet, USB-muistitikku ja puhelin. Koulutusmallissa käytettiin yli 200 kuvaa varmistamaan, että malli oppii tunnistamaan eri esineet erilaisissa kuvakulmista.

Kuvien käsittelyssä ja merkitsemisessä ennen koulutusmallia kuviin merkittiin käsin neliötä Label Studio -ohjelmaa käyttäen (Kuva 2). Laitteen kouluttamisessa käytetyt esineet rajattiin kuvissa ja lajiteltiin oikeisiin luokkiin. Huolellinen kuvien merkitseminen varmistaa, jotta konejakojärjestelmä tunnistaisi esineet liukuhihnalta tarkasti. Merkintöjen jälkeen kuvat jaettiin koulutus-, validointi- ja testijoukkoihin, jotta konenäössä on hyvä ja tarkka tunnistaminen esineille liukuhihnalla.

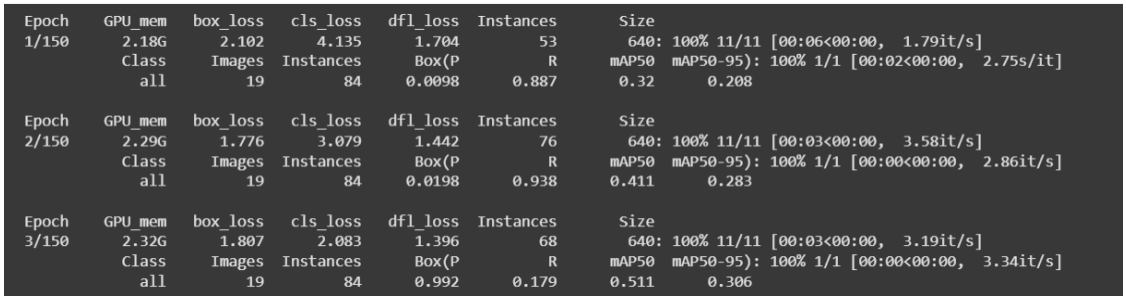


Kuva 2. Kuvassa merkitään ja luokitellaan eri esineitä Label Studio -ohjelmassa, jonka jälkeen kuvilla koulutetaan konenäkömalli.

Koulutusmallin opetus tapahtui Google Colabissa, Ultralyticsin YOLOv5-mallia käyttäen, joka on tarpeeksi kevyt ja tehokas toimiakseen Raspberry Pi 4 -mini-tietokoneessa. Google Colabissa ladattiin Ultralytics-kirjasto, joka on avoimen-lähdekoodin Python-kirjasto. Sen avulla ladattiin YOLOv5-syväoppimismalli, joka on kevyt ja tarpeeksi suorituskykyinen konenäkömalli. YOLOv5 on suunni-teltu objektien ja kuvien tunnistukseen [20].

Konenäkömallin opetusvaiheessa mallia koulutettiin tunnistamaan ja luokittele-maan kuvat 150 epochin verran (Kuva 3). Epoch tarkoittaa yhtä sykliä, jossa

katsotaan kaikki alkuperäiset ja neliöllä merkityt kuvat esineistä. Jokaisen epochin jälkeen syväoppimismalli yrittää parantaa esineiden tunnistus tarkkuutta vertailemalla sen ennusteita oikeisiin tuloksiin. Epocheja pitää olla sopiva määrä, jotta malli toimii oikein. Jos epocheja on liian monta, se voi aiheuttaa ylisovittamisen, joka tarkoittaa sitä, että malli oppii liikaa esineiden yksityiskohtia eikä voi tunnistaa uusia kuvia tarkasti. Jos epocheja on liian vähän, mallin kuvantunnistus kyky heikkenee.



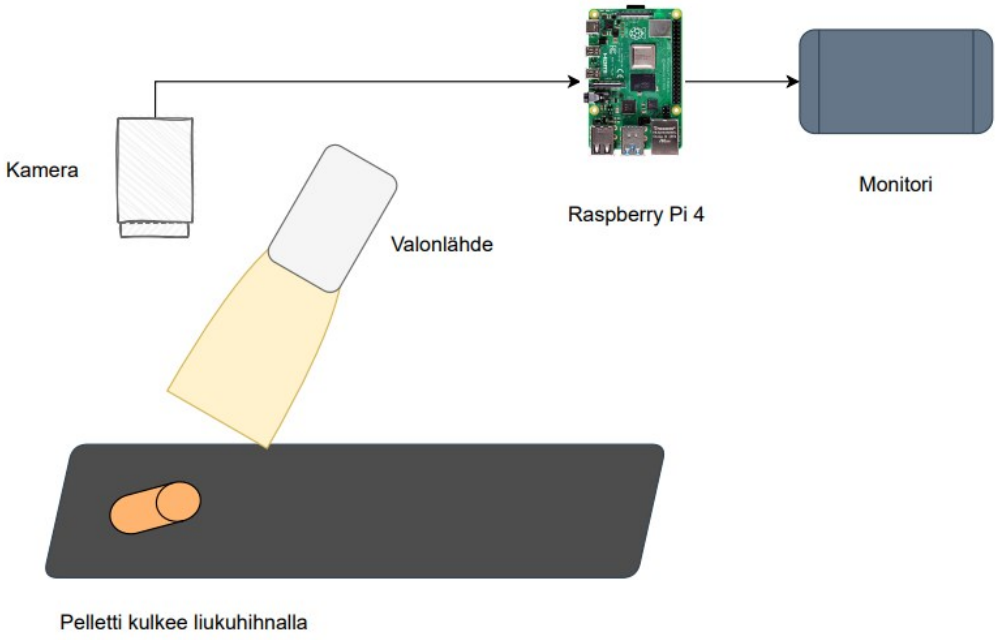
```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
1/150      2.18G      2.102      4.135      1.704         53        640: 100% 11/11 [00:06<00:00,  1.79it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:02<00:00,  2.75s/it]
             all         19         84     0.0098      0.887       0.32      0.208

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
2/150      2.29G      1.776      3.079      1.442         76        640: 100% 11/11 [00:03<00:00,  3.58it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.86it/s]
             all         19         84     0.0198      0.938       0.411      0.283

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
3/150      2.32G      1.807      2.083      1.396         68        640: 100% 11/11 [00:03<00:00,  3.19it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  3.34it/s]
             all         19         84      0.992      0.179       0.511      0.306
```

Kuva 3. Konenäkömallin epoch-syklit Google Colabissa, jossa mallin tarkkuus paranee jokaisen syklin jälkeen.

3 Konenäköjärjestelmän laitteet ja ohjelmistot

3.1 Konenäköjärjestelmän suunnittelu

Tässä opinnäytetyössä suunniteltiin ja kehitettiin konenäköjärjestelmä niin, että Raspberry Pi 4 -minitietokone asetettiin muovisen laatikon päälle, joka sijaitsee pellettikoneen liukuhihnan yläpuolella (Kuva 4). Minitietokone sitten yhdistettiin viereisellä pöydällä olevaan tietokoneen näyttöön sekä langalliseen hiireen ja näppäimistöön. Virtalähteenä käytettäisiin 5 voltin USB-seinälaturia (Kuva 5).



Kuva 4. Konenäköjärjestelmän suunnittelu kuva ennen projektin aloittamista .

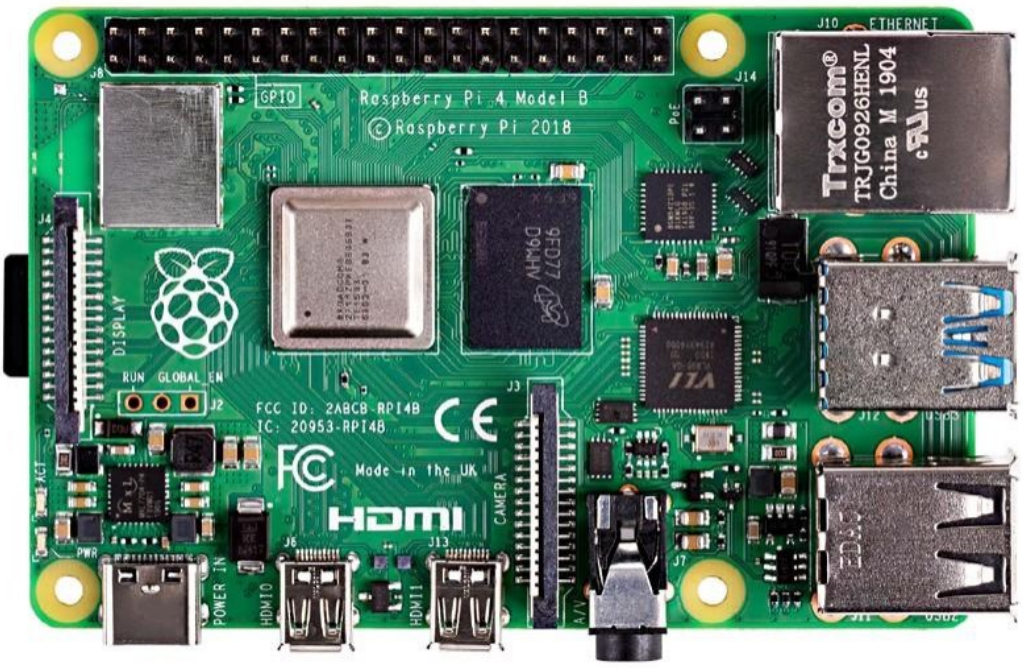
Kuva 5. Energy+ 5V USB-seinälaturi, joka toimi Raspberry Pi 4 -minitietokoneen virtalähteenä.

3.2 Raspberry Pi 4 -minitiekoneen historia ja esittely

Tässä insinöörityössä käytettiin Raspberry Pi 4 -minitietokonetta pellettikoneen konenäköjärjestelmän toteutukseen (Kuva 6). Ensimmäinen Raspberry Pi julkaistiin vuonna 2012. Sen valmisti Raspberry Pi Foundation, joka on voittoa tavoittelematon järjestö Cambridgessa, Englannissa. Järjestön tavoitteena on kannustaa tietotekniikan opetusta kouluissa. [14]

Raspberry Pi 4 -minitietokone sisältää neliytimisen ARM Cortex-A72-prosessorin ja 1 gigatavua RAM-muistia. Laiteessa on kaksi micro-HDMI-porttia näyttöjen yhdistystä varten ja neljä USB-porttia. Verkkoyhteyden voi muodostaa langallisesti Ethernet-kaapelilla tai langattomasti Wi-Fi-yhteydellä. Raspberry Pi 4:ssä on myös 40-pinninen GPIO-liitin, mihin voi yhdistää erilaisia antureita, moottoreita ja muita sähköisiä komponentteja. Laitteen virtalähteenä voi käyttää USB-C-virtalähdettä esimerkiksi tietokoneen USB-porttia tai matkapuhelimen laturin muuntajaa, jossa on USB-portti [14].

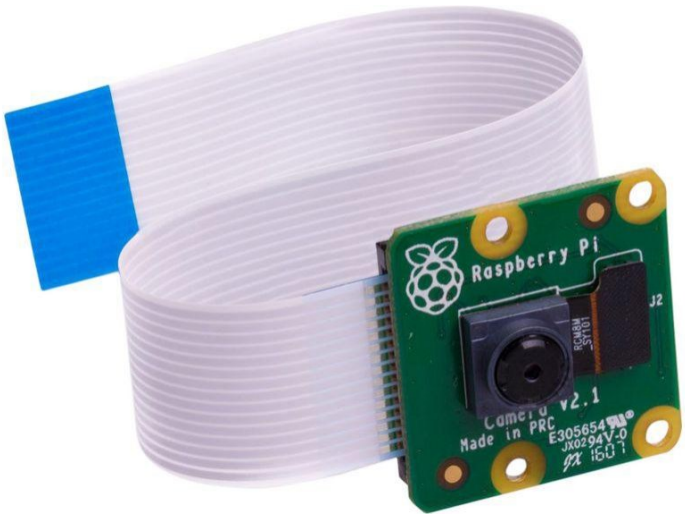
Raspberry Pi 4 -minitietokone käyttää Debian-pohjaista käyttöjärjestelmää nimeltä Raspberry Pi OS. Käyttöjärjestelmä asennetaan MicroSD-muistikortille ja se liitetään minitietokoneeseen. Lisäksi se tukee monia eri ohjelmointikieliä ja työkaluja [14].



Kuva 6. Konenäköjärjestelmän alusta Raspberry PI 4 -minitietokone.(Partco, 2023.)

3.2.1 Raspberry Pi Camera -moduuli

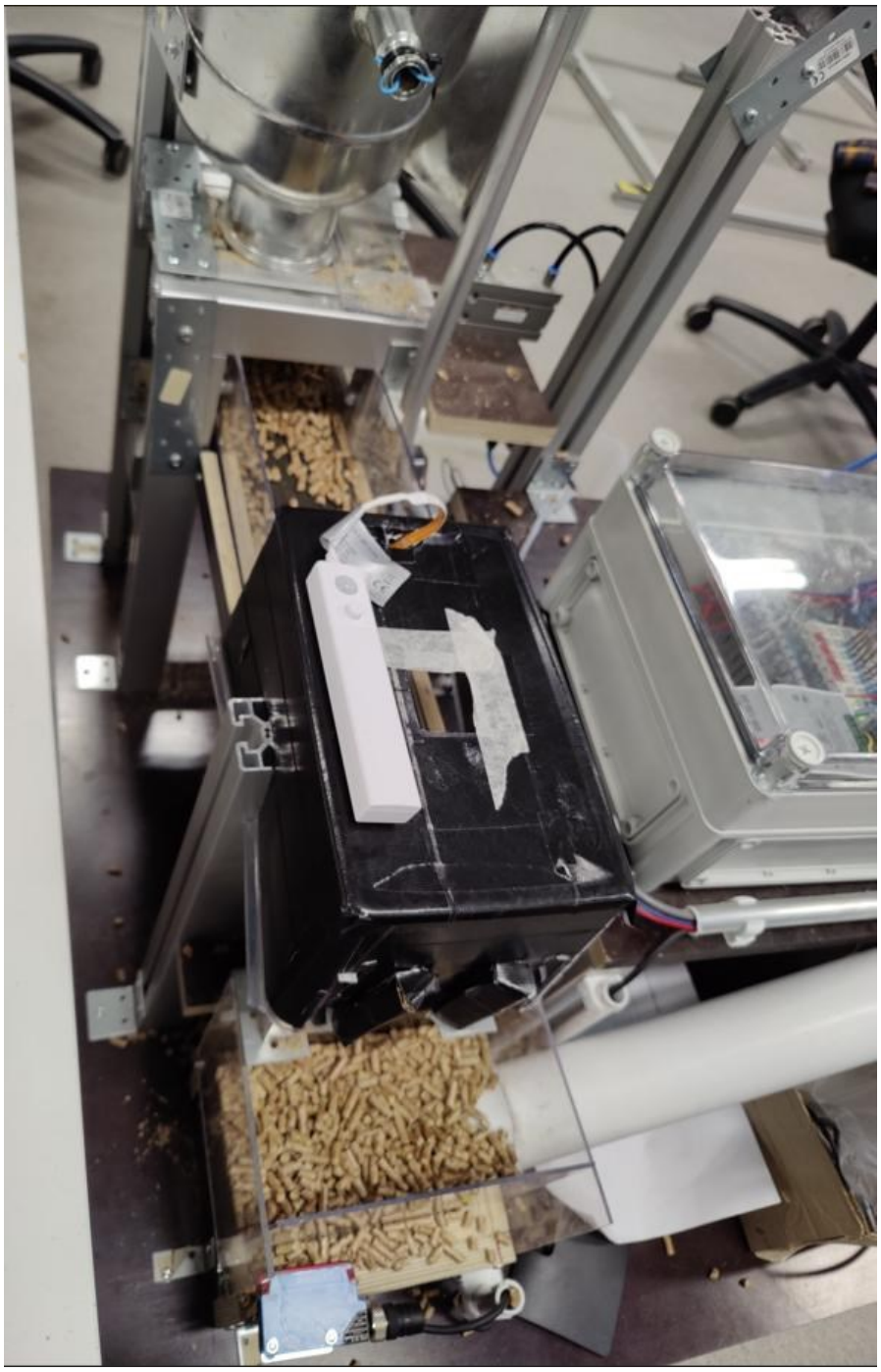
Raspberry Pi Camera V2 -moduulia käytettiin tämän konenäköjärjestelmän kamerana (kuva 7). Kamerassa on 8 megapikselin Sony IMX219 -kuvakenno ja se liitetään minitietokoneeseen CSI-nauhakaapelilla (Camera Serial Interface). Kameralla voi ottaa stillkuvia 4k resoluutiolla ja tallentaa videoita Full HD -tarkkuudella. Kameraa voi ohjata Python -ja C-kielisillä kirjastoilla, kuten Picamera ja OpenCV. [15]



Kuva 7. Raspberry Pi Camera Module V2. (Raspberry Pi, 2023.)

3.2.2 Konenäköjärjestelmän valaistus

Konenäköjärjestelmän valaistuksena käytettiin muovista laatikkoa, jonka pohjaan oli teipattu led-valonauhaa (Kuva 8 ja Kuva 9). Laatikon oli valmiiksi koululla, ja sen oli tehnyt aiemmin toinen oppilas. Laatikon pohjaan oli tehty kaksi reikää, joista isompi oli kameralle ja pienempi led-valonauhalle. Lisäksi laatikon sivuja oli leikattu, jotta se sopisi paremmin liukuhihnan päälle. Laatikko oli peitetty mustalla teipillä, mikä auttoi vähentämään varjoja kameran kuvissa. Led-valonauhan virtalähteenä käytettiin neljää AAA-paristoa.



Kuva 8. Konenäköjärjestelmän valaistuslaatikko.

Kuva 9. Valaistuslaatikon sisäpuoli.

3.3 Python-ohjelmointikielen esittely

Python on monipuolinen ja helppokäyttöinen ohjelmointikieli, jota käytetään eri sovelluksissa, kuten konenäössä, automaatiossa ja tietojenkäsittelyssä. Python-ohjelmointikielen kehitti Guido van Rossum ja se julkaistiin vuonna 1991. Ohjelmointikieli on suunniteltu selkeäksi ja helposti luettavaksi. [12]

Python tukee useita ohjelmointityylejä, joista yleisimpiä on oliopohjainen ohjelmointi ja proseduraalinen ohjelmointi. Oliopohjaisessa ohjelmoinnissa koodi jaetaan luokkiin ja olioihin, joiden avulla voidaan hallita monimutkaisia järjestelmiä helpommin. Tämä on hyödyllistä silloin, kun kehitetään laajoja ohjelmistoja tai moduuleja, joita voidaan käyttää uudelleen eri projekteissa. [12]

Proseduraalisessa ohjelmoinnissa taas koodi kirjoitetaan erillisiin funktioihin, jotka suorittavat tiettyjä tehtäviä. Ohjelmointityyli on yksinkertainen ja soveltuu hyvin pieniin ohjelmiin, joissa halutaan suorittaa tietyt tehtävät peräkkäin ilman monimutkaista rakenteellista jaottelua. [13]

Python-ohjelmointikielelle on laaja kirjastovalikoima, joka mahdollistaa sovellusten kehittämisen ilman, että tarvitsee ohjelmoida kaikkea alusta asti. OpenCV-kirjasto on yksi yleisimmin käytetyistä työkaluista konenäön toteutuksessa ja se sisältää valmiita toimintoja esimerkiksi kuvankäsittelyyn, objektien tunnistamiseen ja koneoppimisen hyödyntämiseen visuaalisessa datassa. [10]

3.3.1 Konenäköjärjestelmän ohjelma

Konenäköjärjestelmän ohjelmiston pohjana käytettiin Core Electronicsin valmista YOLO-konenäkömalliin perustuvaa koodipohjaa. Koodia piti muokata kevyemmäksi pienentämällä resoluutiota ja optimoimalla kuvankäsittelyä. Koodissa ensin kamera alustetaan luomalla Picamera2 -kirjaston avulla ja sille määrätään resoluutioksi 320x320 suorituskyvyn parantamiseksi. Sen jälkeen ohjelma lataa YOLOv5-konenäkömallin, joka on valmiiksi koulutettu tunnistamaan puiset pelletit ja mahdolliset vierasesineet. [3]

Konenäköohjelma toimii jatkuvassa esineiden tunnistus syklissä, jossa PiCamera ottaa kuvia ja YOLOv5-konenäkömalli tarkastelee ne tunnistaakseen esineet samanaikaisesti. Videokuvasyötteessä tunnistettujen esineiden päälle piirretään rajauslaatikoita ja ne nimetään opetettuihin luokkiin. Ohjelma sammutetaan painamalla Q-näppäintä. [3]

```
import cv2
from picamera2 import Picamera2
from ultralytics import YOLO
```

Esimerkkikoodi 1. Ohjelman ensimmäisessä vaiheessa tuodaan tarvittavat kirjastot. Cv2 (OpenCV) vastaa kuvien esittämistä ja tekstin piirtämistä, picamera2 komennolla hallitaan Raspberry Pi:n kameramoduulia ja ultralytics.YOLO komento mahdollistaa YOLOv5-konenäkömallin lataamisen ja käyttämisen.

```
picam2 = Picamera2()
picam2.preview_configuration.main.size = (640, 640)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()
```

Esimerkkikoodi 2. Konenäköjärjestelmän kamera alustetaan Picamera2-kirjastolla. Kuvan resoluutioksi asetetaan 640x640 pikseliä ja kuvan väri formaatiksi valitaan RGB888. Kameran konfigurointi tapahtuu esikatselutilassa (preview) ja lopuksi kamera käynnistetään (picam2.start).

```
model = YOLO("/home/elias/Desktop/yolo_object_detec-
tion/uusi my model/my_model_updated_ncnn_model")
```

Esimerkkikoodi 3. Tässä vaiheessa ladataan valmiiksi koulutettu YOLOv5-konenäkömalli, joka on muutettu NCNN-formaattiin.

```
while True
        frame = picam2.capture_array()
```

Esimerkkikoodi 4. Tässä vaiheessa ohjelma siirtyy jatkuvaan silmukkaan, jossa jokaisella suorituskerralla otetaan uusi kuva kameralla.

```
results = model(frame, imgsz=320)
annotated_frame = results[0].plot()
```

Esimerkkikoodi 5. Otetulle kuvalle suoritetaan esinetunnistus YOLO-mallilla. Malli palauttaa havaitut kohteet, sekä luottomusarvot. Plot()-komento piirtää havaitut kohteet kuvan päälle.

```
inference_time = results[0].speed['inference']
fps = 1000 / inference_time
```

Esimerkkikoodi 6. Tunnistuksen suoritusaika mitataan ja muunnetaan kuvataajuudeksi (FPS).

```
text = f'FPS: {fps:.1f}'
font = cv2.FONT_HERSHEY_SIMPLEX
text_size = cv2.getTextSize(text, font, 1, 2)[0]
text_x = annotated_frame.shape[1] - text_size[0] - 10
text_y = text_size[1] + 10
cv2.putText(annotated_frame, text, (text_x, text_y), font,
1, (255, 255, 255), 2, cv2.LINE_AA)
```

Esimerkkikoodi 7. FPS-arvo piirretään kuvan oikeaan yläkulmaan.

```
cv2.imshow("Camera", annotated_frame)
if cv2.waitKey(1) == ord("q"):
break
```

Esimerkkikoodi 8. Merkitty kuva näytetään ruudulla OpenCV:n imshow-funktiolla. Ohjelma pysyy silmukassa, kunnes painetaan Q- näppäintä.

```
cv2.destroyAllWindows()
```

Esimerkkikoodi 9. Kun silmukka päättyy, kaikki avoimet Opencv-ikkunat suljetaan.

3.3.2 NCNN muunnos formaatti

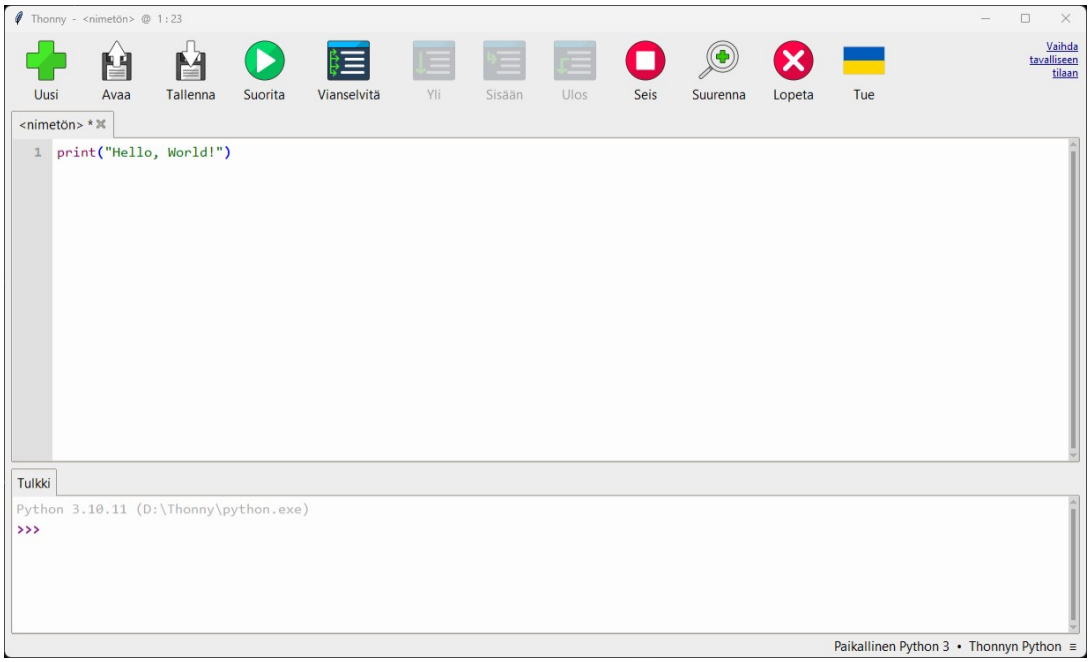
Alkuperäisessä konenäköpohjassa käytettiin Pytorch-formaattia ja se oli Raspberry Pi 4:lle liian raskas, niin konenäkömalli muutettiin kevyempään NCNN-formaattiin [3]. Formaatin muutokseen piti ladata vanhempi Pytorch versio, jotta Raspberry Pi 4 ei kaatuisi muutoksen aikana. NCNN-formaatin avulla konenäkömallin FPS (frames per second) kaksinkertaistui ilman, että tarkkuus tai tunnistusteho heikentyi.

```
from ultralytics import YOLO
model = YOLO("/home/elias/Desktop/yolo_object_detec-
tion/uusi my model/my_model_updated.pt")
model.export(format="ncnn", imgsz=320)
```

Esimerkkikoodi 10. Ohjelmassa valmiiksi koulutettu YOLOv5-malli muunnetaan Pytorch-formaatista NCNN-formaatiin ja luodaan uusi kevyempi YOLOv5-malli.

3.3.3 Thonny -ohjelmointiympäristön esittely

Thonny on yksinkertainen ja käyttäjäystävällinen Python-ohjelmontiympäristö, joka on kehitetty aloittelijoille ja sen kehitti virolainen ohjelmoija Aivar Annamaa. Thonnyssa on sisäänrakennettu Python-tulkki, joka mahdollistaa ohjelmoinnin ilman erillisiä asennuksia. Ohjelmointiympäristössä on myös kehittyneitä ominaisuuksia, kuten erivaiheiden mukaan edistyvä debuggeri eli virheitä etsivä ohjelmointityökalu, visuaalinen muuttujien tarkastelu sekä automaattinen sievennysten ja syntaksin tarkistus. Thonny on yksi Python -ohjelmointikielen monipuolisimmista IDE-editoreista (Kuva 10). [19]



Kuva 10. Esimerkkikuva Thonny-ohjelmointiympäristöstä.

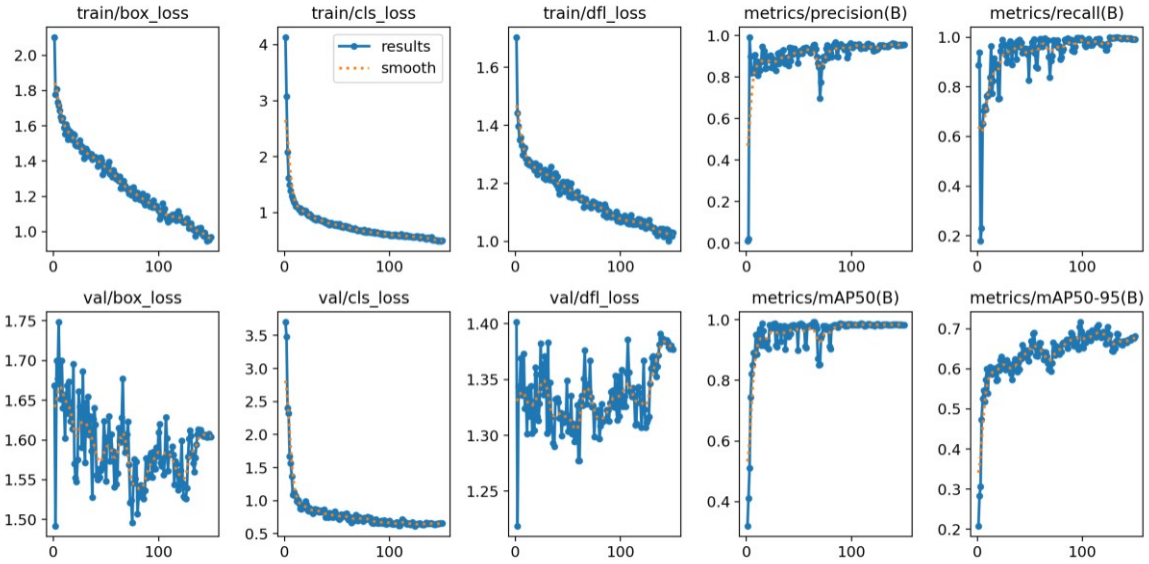
3.3.4 Google Colab -ympäristön hyödyntäminen konenäön koulutuksessa

Google Colab on ilmainen pilvipohjainen kehitysympäristö, joka kehitettiin vuonna 2017. Sen avulla voi ohjelmoida python-kielellä helposti verkkoselaimessa ilman muita asennuksia. Colabissa on mahdollista käyttää GPU-resursseja (näytönohjaimia) ja TPU:ita (Tensor Processing Unit), jotka ovat hyödyllisiä projekteihin, jossa tarvitaan suuria näytönohjaimen laskentatehoja esimerkiksi konenäön koulutuksessa. [6]

Google Colabissa voi käyttää valmiita ohjelmointiskriptejä, kuten ”Train YOLO Models in Google Colab” (tekijänä Evan Juras, EJ Technology Consultants), joiden avulla voi kouluttaa konenäköä Google Colabin pilvipalvelussa. Tässä opinnäytetyössä käytettiin Evan Jurasin kehittämää Python -ohjelmaa konenäön kouluttamiseen. [5]

Kuvassa 11 on esitetty koulutusprosessin suorituskykymittareita ja virhefunktioiden (loss) kehitystä konenäkömallissa. Kuvaajan ylärivi esittää koulutusdatan (train) aikana mitattuja arvoja. Kuvaajan alarivi näyttää validointidatan (val) vastaavat mittarit. Kuvassa taulukoiden vaaka-akseli kuvaa koulutuksen kehitystä epocheissa, joista jokainen merkitsee yhtä kokonaista koulutusdatan läpikäyntiä. Pystyakselilla kuvataan kyseisen mittarin tai virhefunktion arvoa. tämä kertoo, kuinka hyvin konenäkömalli oppii ja parantaa tarkkuuttaan koulutuksen edetessä.

Kuten kuvassa 11 nähdään, konenäkömallin tarkkuus validointidatalla paranee aluksi epochien edetessä ja alkaa tasoittua koulutusjakson lopussa. Tämä viittaa siihen, että konenäkömalli on oppinut suurimman osan datasta eikä lisäkoulutus paranna sen suorituskykyä ja tarkkuutta.



Kuva 11. Konenäkömallin koulutuksen opetuskäyrät.

3.4 YOLOv5-konenäkömallin käyttö ja soveltaminen

YOLO (You Only Look Once) on yksi tunnetuimmista ja tehokkaimmista konenäkömalleista, jota käytetään esineiden objektitunnistuksessa ja -luokittelussa. YOLO-mallit ovat nopeita ja tarkkoja. Ne pystyvät suorittamaan koko objektin tunnistusprosessin yhdellä suorituskierroksella. YOLO-mallit toimivat siten, että ne jakavat kuvan ruudukkoon ja ennustaa objektien sijainnit ja luokat [20].

Uusin versio YOLO-mallista on YOLOv11, mutta tässä opinnäytetyössä käytettiin YOLOv5-mallia, koska se on tarpeeksi kevyt ja tehokas Raspberry Pi 4-minitietokoneelle. YOLOv5 on optimoitu useille laitteille ja se mahdollistaa hyvän suorituskyvyn ja tarkkuuden pienillä resursseilla. [20]

YOLO-konenäkömalleissa voi käyttää myös valmiita kirjastoja esimerkiksi COCO-kirjastoa (Common Objects in Context). COCO on laajasti käytetty objekti- ja segmentointidatasetti, joka sisältää 330 000 kuvaa ja 80 erilaista objektikategoriaa. COCO-datasetissä on valmiiksi luokiteltuja ja merkittyjä kuvia, jotka mahdollistavat tehokkaan mallin koulutuksen. COCO-kirjastoa voidaan käyttää yhdessä useiden konenäkömallien kanssa ja sitä voi käyttää mallien esikoulutuksessa tai omien mallien kehityksessä. [2]

3.5 Koulutettavien esineiden kuvienkäsittely

Tässä opinnäytetyön luvussa käsitellään koulutettavien esineiden kuvienkäsittelyä ja esikäsittelyn merkitystä konenäkömallin tarkkuuden ja luotettavuuden kannalta. Konenäkömallin tarkkuus ja luotettavuus perustuu koulutuksessa käytetystä kuvadatasta. Kuvien laatu, monipuolisuus ja ennakkokäsittely vaikuttavat konenäkömallin suorituskykyyn eri ympäristöissä. Kuvien esikäsittelyssä esineiden huolellinen rajaaminen on erittäin tärkeää, koska se varmistaa, että konenäkömalli oppii tunnistamaan esineet tarkasti ilman häiriöitä taustasta tai muista

objekteista. Lisäksi tärkeää on se, että opetuskuvissa olevat esineet ovat selkeästi erillään toisista esineistä. Jos esineet ovat liian lähellä toisia esineitä, se vaikeuttaa esineiden tarkkaa rajaamista.

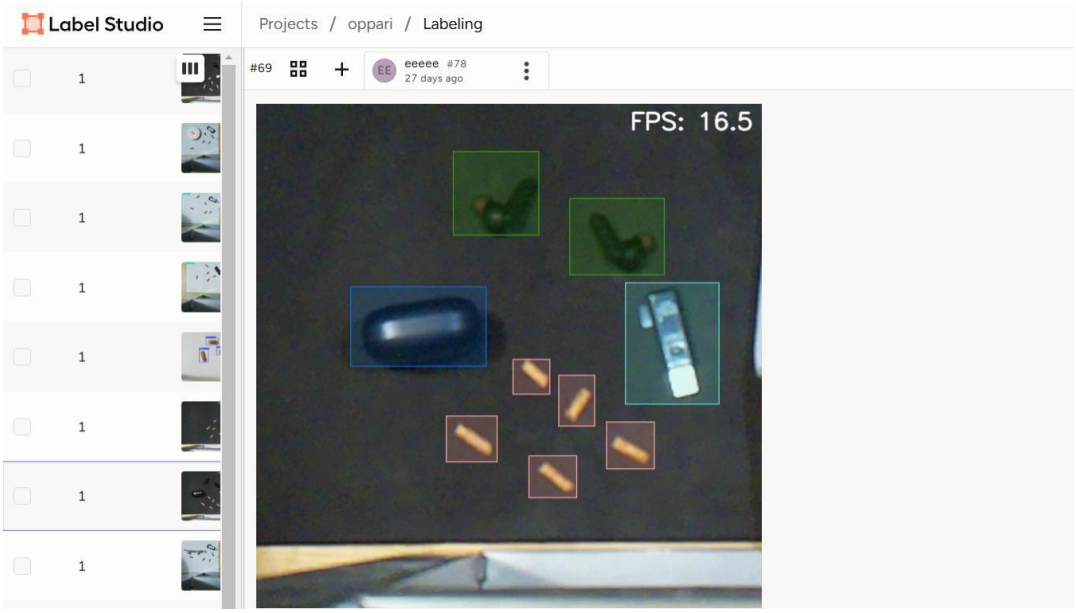
Konenäkömallin kuvien ottamisessa käytettiin eri valaistusolosuhteita mallin tarkkuuden parantamiseksi. Jos opetuskuvat ovat liian kirkkaita tai tummia, se heikentää mallin kykyä erottaa yksityiskohtia ja vaikuttaa esineiden tunnistustarkkuuteen.

Kaikki opetuskuvat otettiin ylhäältä alaspäin ja esineiden asentoa vaihdeltiin eri kuvissa. Kuvien taustoina käytettiin valkoista paperia ja pellettikoneen mustaa liukuhihnaa. Opetuskuvien resoluutiona käytettiin 640x640, sillä se on tarpeeksi tarkka yksityiskohtien säilyttämiseen ja se toimii hyvin Google Colabin GPU:n laskentateholla.

3.5.1 Opetuskuvien merkintä ja luokittelu Label-Studio -ohjelmalla

Tässä luvussa esitellään, miten konenäkömallin opetuskuvat käsiteltiin ja luokiteltiin. Kun konenäkömallin opetuskuvat oli otettu, ne siirrettiin Label-Studio -ohjelmaan, jossa objektien rajaukset merkittiin ja luokiteltiin. Kuvia oli noin 200 ja niissä oli puisia pellettejä ja esimerkillisiä vääriä esineitä, kuten puhelin, nappikuulokeet ja USB-muistitikku. Kun kuvat oli merkitty Label-Studio -ohjelmaan, saatiin neljä erillistä tiedostoa. Ensimmäisessä tiedostossa oli alkuperäiset kuvat, toisessa rajatut kuvat, kolmannessa lista opetetuista esineistä ja neljännessä kategoria lista merkityistä esineistä.

Label-Studio on avoimen lähdekoodin työkalu datan merkintään ja luokitteluun. Se valittiin tähän konenäköjärjestelmän koulutukseen, koska se on ilmainen ja helppo käyttää (Kuva 12).

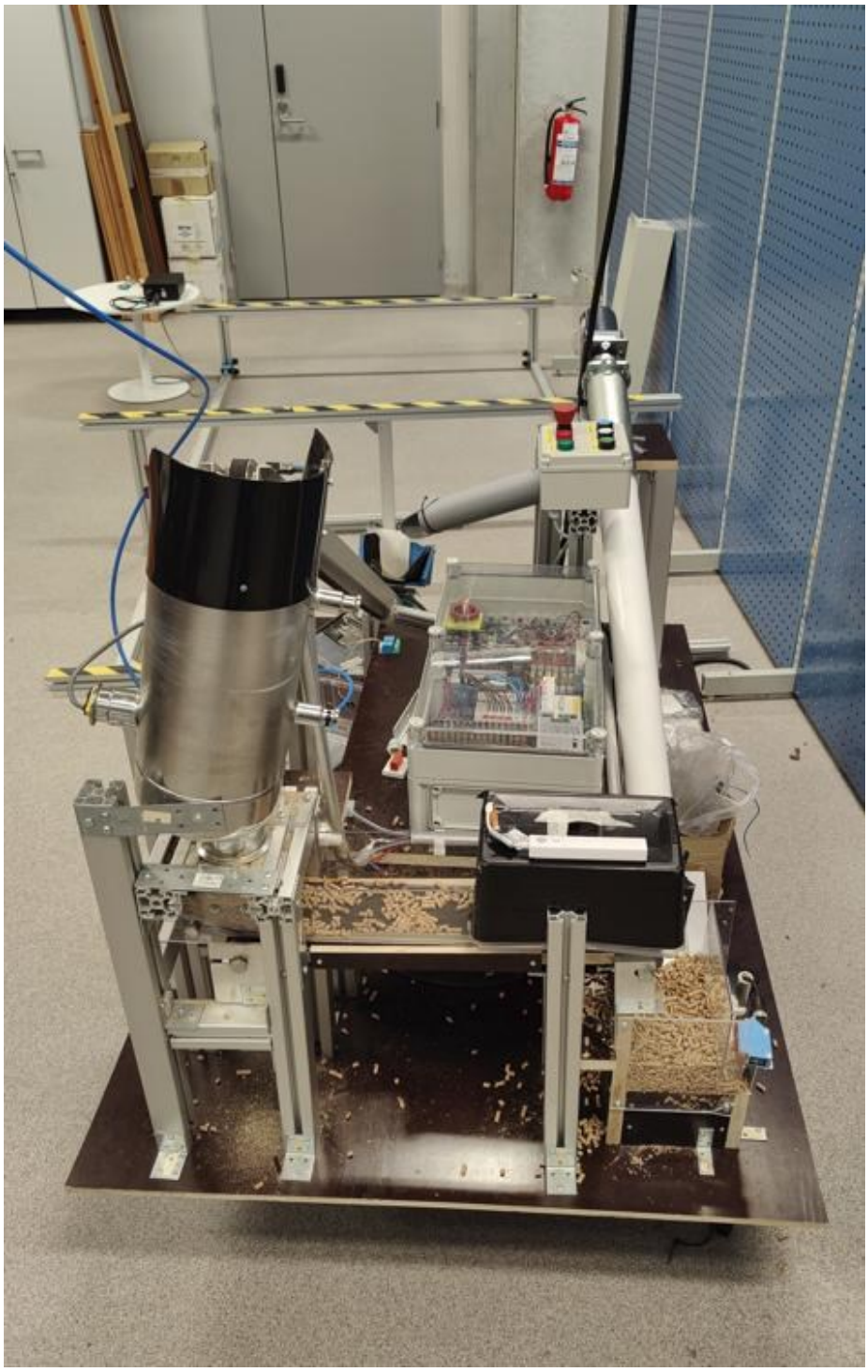


Kuva 12. Esimerkkikuva Label-Studio -ohjelmasta.

4 Pellettikuljettimen esittely

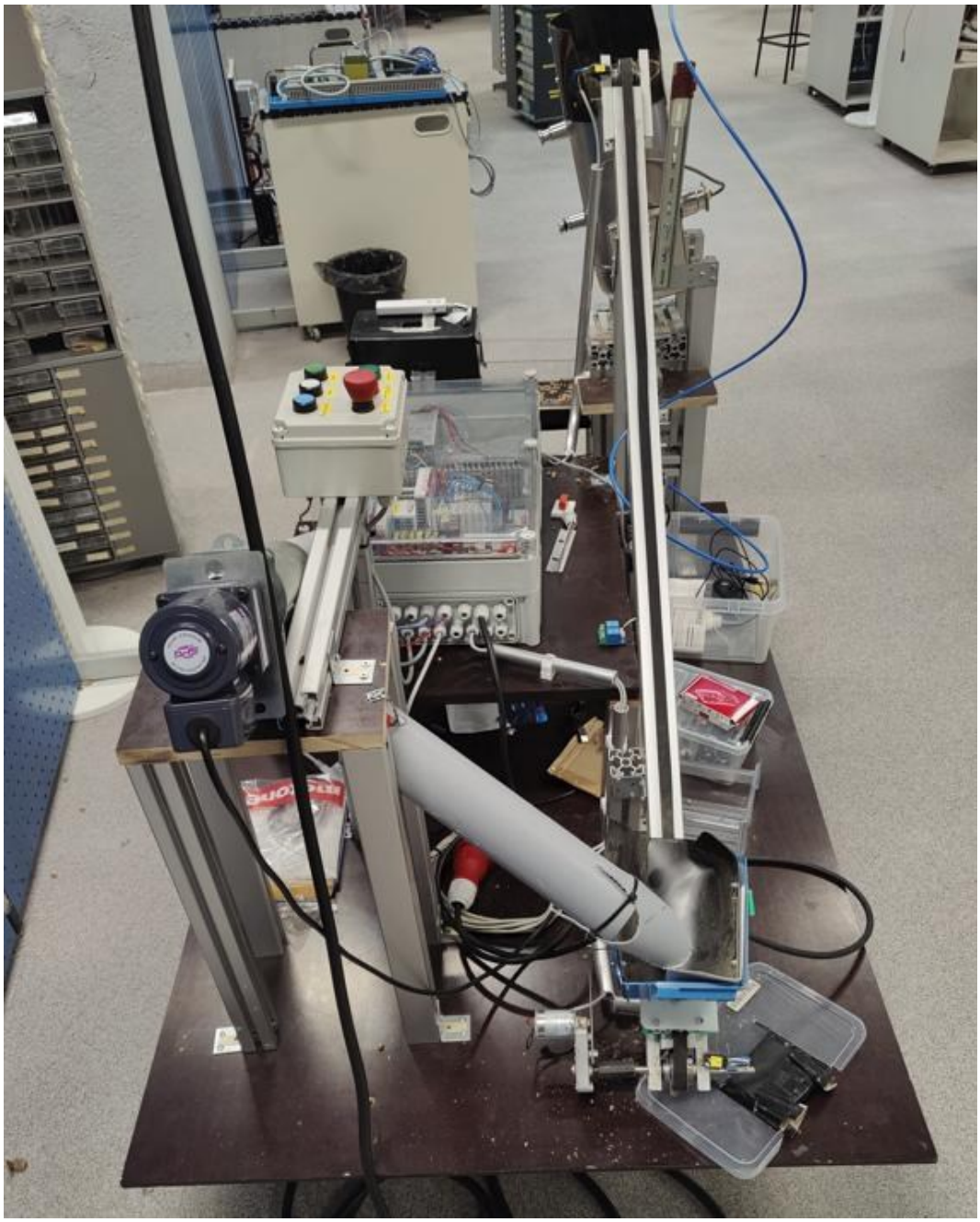
Konenäköjärjestelmä tehtiin Metropolia Ammattikorkeakoulun Myyrmäen kampuksen automaatiolaboratoriossa sijaitsevaan pellettikuljettimeen ja sen toi-minta perustuu kiertävään prosessiin, jossa pelletit kulkevat kahden muovisen säiliön välillä. Aluksi pelletit tippuvat yläsäiliöstä liukuhihnalle pneumaattisen luukun avulla ja liukuhihnan yläpuolella oleva konenäköjärjestelmä tarkkailee pellettejä ja muita mahdollisia ei haluttuja esineitä. Sen jälkeen liukuhihna kuljet-taa pelletit alasäiliöön, josta ne siirtyvät ruuvinkuljettimeen ja sen jälkeen hissiin, joka kuljettaa pelletit takaisin yläsäiliöön (Kuva 13 ja Kuva 14).

Pellettikuljettimen eri osien toimintaa ohjataan ohjelmoitavalla logiikalla, joka on asetettu kuljettimen keskellä olevaan sähkökeskukseen. Pellettikuljetin saa käyttövirtansa kolmivaiheisesta voimapistorasiasta (Kuva 15).



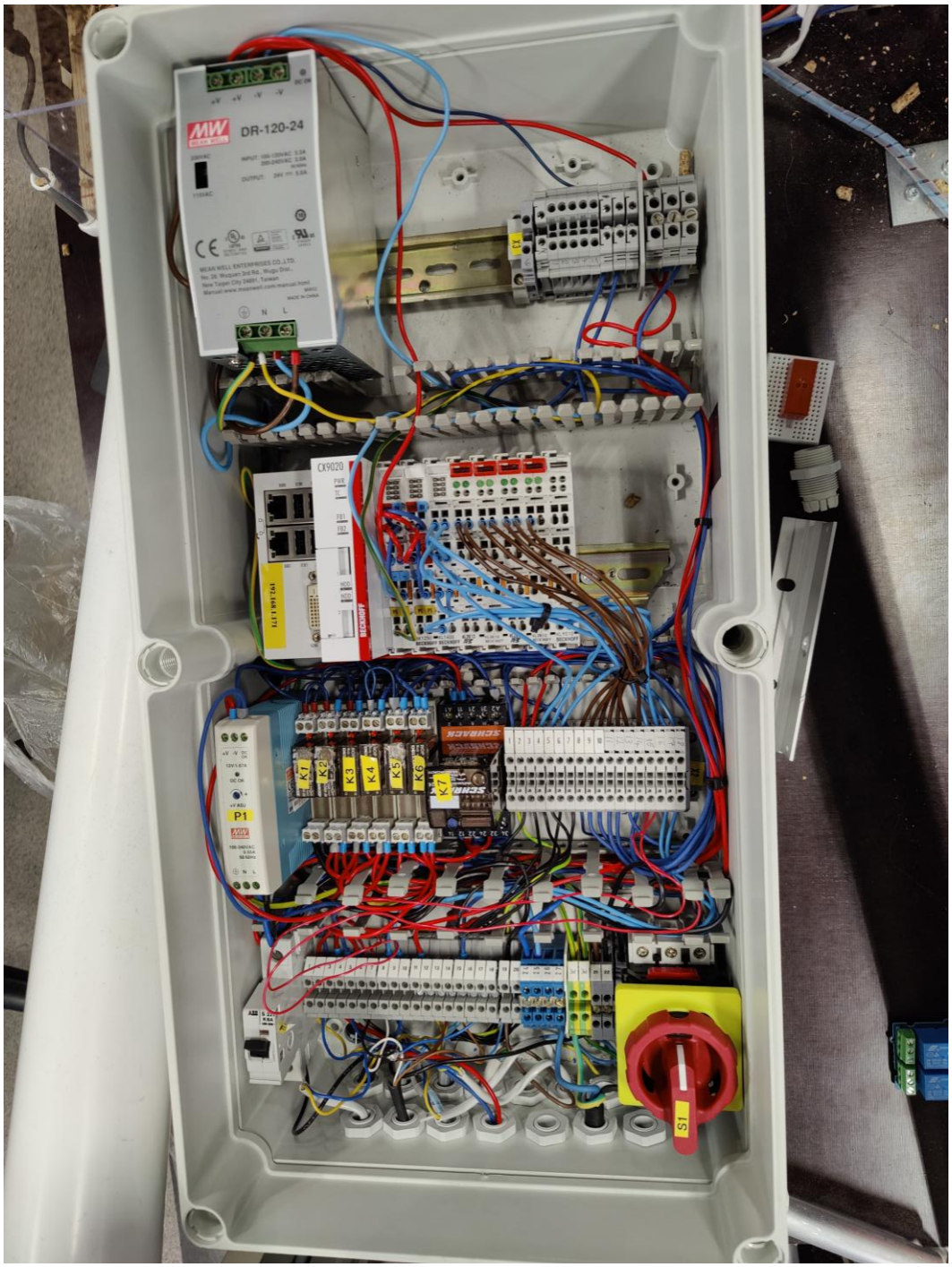
Kuva 13. Pellettikuljetin, mihin konenäköjärjestelmä tehtiin.

Kuvassa 13 näkyy prosessi, jossa vasemmalla olevasta yläsäiliöstä tippuu pellettejä liukuhihnalle, missä konenäköjärjestelmä valvoo sitä ja sen jälkeen pelletit tippuvat alasäiliöön, jossa ne siirtyvät ylös ruuvinkuljettimella.



Kuva 14. Pellettikuljettimen toinen puoli.

Vasemmalla kuvassa 13 näkyy ruuvikuljetin, josta pelletit siirtyvät hissiin, joka kuljettaa pelletit ylös takaisin yläsäiliöön prosessin kiertoa varten.

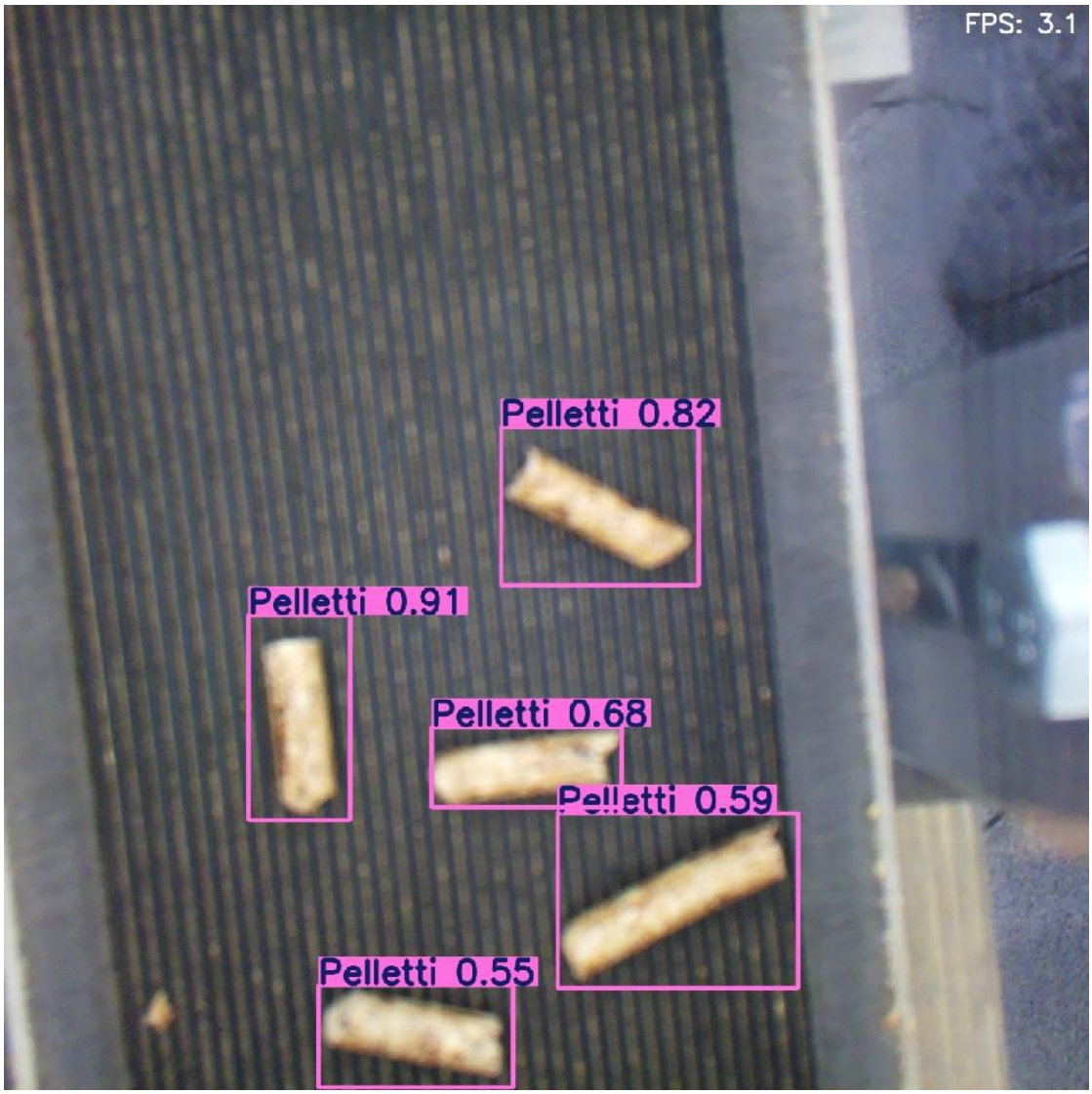


Kuva 15. Pellettikuljettimen ohjelmoitavan logiikan sähkökeskus.

5 Konenäköjärjestelmän toimivuuden testaaminen

Konejärjestelmän testausta varten Metropolia ammattikorkeakoulun pellettikuljetin siirrettiin tietokone pöydän viereen, jotta Raspberry Pi 4 -minitietokone pystyttiin yhdistämään tarpeellisiin laitteisiin (Kuva 16). Sitten Raspberry Camera aseteltiin liukuhihnan yllä olevan valaistuslaatikon päälle. Konenäkö käynnistettiin ja huomattiin, että Raspberry Pi 4 ei ole tarpeeksi tehokas tunnistamaan isoja kasoja pellettejä tai seuraamaan niitä liikkeessä alhaisen FPS:än takia (Kuva 17). Yksittäiset paikallaan olevat pelletit ja mahdolliset ei-toivotut esineet se tunnisti kohtalaisen tarkasti.

Kuva 16. Konenäköjärjestelmä tarkastelee pellettejä liukuhihnan päällä.



Kuva 17. Konenäkömallin videosyöte, jossa näkyy pelletit ja oikealla ylhäällä näkyy heikko FPS.

Hetkellisen käytön jälkeen huomattiin myös, että minitietokone ylikuumenee liiallisen käytön aikana ja siksi se piti sammuttaa hetkeksi. Ylikuumentamisen voisi välttää liittämällä tuulettimen Raspberryn GPIO liittimeen (Kuva 18).

Kuva 18. Raspberry Pi -tuuletin. (The Pi Hut, 2023)

6 Yhteenveto

Tässä opinnäytetyöprosessissa suunniteltiin ja kehitettiin konenäköjärjestelmä Metropolia Ammattikorkeakoulun pellettikuljettimeen, jonka tehtävänä oli tunnistaa puisia pellettejä ja vieraita esineitä liukuhihnalta. Konenäköjärjestelmä tehtiin Raspberry Pi 4 -minitietokoneella, ja ohjelmoinnissa käytettiin Thonny-nimistä Python-ohjelmointiympäristöä. Konenäkömallina käytettiin Ultralyticsin YOLOv5-mallia. Sen koulutuksessa käytettiin noin 200 itse otettua kuvaa. Kuvien koulutuksessa käytettiin Label-Studio -ohjelmaa, jossa kuvat merkittiin ja luokiteltiin sopiviin luokkiin. Sen jälkeen merkityt kuvat siirrettiin Google Colab -

pilvipalveluun, jossa opetusvaiheessa kuvat käytiin läpi 150 opetuskertaa (epochia).

Konenäkömallin koulutuksen jälkeen konenäkömalli muunnettiin PyTorch-formaatista NCNN-formaattiin, koska se paransi konenäkömallin suorituskykyä. Konenäköjärjestelmä aseteltiin pellettikuljettimen liukuhihnan yläpuolelle, johon oli teipattu muovinen laatikko varjojen vähentämiseksi. Laatikon sisällä oli teipattu led-valonauhaa valaistukseksi, jotta videokuva olisi selkeämpi.

Lopputuloksena todettiin, että Raspberry Pi 4 -minitietokone ei ole tarpeeksi tehokas käyttämään konenäköohjelmaa, jos puiset pelletit liikkuvat liukuhihnalla alhaisen kuvataajuuden takia. Liukuhihnalla yksittäisten paikalla olevien pellettien ja vieraiden esineiden tunnistus onnistui melko tarkasti, koska alhainen kuvataajuus riitti siihen.

Opinnäytetyön alussa oli erilaisia haasteita, jotka hidastivat konenäköjärjestelmän toteutusta. Ensimmäinen ongelma liittyi Raspberry Pi 4 -minitietokoneen epävakauteen. Minitietokone sammui välillä ohjelmoinnin aikana ja Chromium -verkkoselainta ei voinut edes käynnistää. Tämä ongelma selvisi sen jälkeen, kun Raspberryn omaa muistintarkistusohjelmaa käytettiin muistikortintarkistukseen. Ohjelma ilmoitti, että muistikortti ei täyttänyt järjestelmän vaatimuksia. Muistikortin tilalle hankittiin uusi muistikortti.

Konenäköjärjestelmän kehittämisprosessin seuraavassa vaiheessa etsittiin sopivan kevyt ja suorituskykyinen konenäkömalli. Useita konenäkömalleja jouduttiin testaamaan, koska Raspberry Pi 4 -minitietokone ei ole tarpeeksi tehokas uusimmille konenäkömalleille. Konenäkömallin valinnan jälkeen ongelmana oli konenäkömallin muunnos PyTorch-formaatista NCNN-formaattiin. Aina kun muunnosta yritettiin tehdä, Python-ohjelma lakkasi toimimasta. Ongelma ratkaistiin lataamalla vanhempia Pytorch-ohjelman päivityksiä. Esimerkiksi mikään Pytorch-ohjelman versio yli 2.5.0 ei toiminut Raspberry Pi 4 -minitietokoneella.

Konenäköjärjestelmää voisi kehittää paremmaksi hankkimalla uusimman version Raspberry Pi -minitiekoneesta 8Gt:n RAM-muistilla ja lisäämällä siihen prosessorintuulettimen. Myös korkealaatuisemman kameran voisi hankkia esimerkiksi Raspberry Pi High Quality Camera -moduulin. Konenäköjärjestelmän konenäkömallina voisi käyttää uusinta YOLO-neuroverkkoa esimerkiksi YOLOv11.

Lähteet

1. CERN. *History of Machine Vision*. Verkkoaineisto: https://cds.cern.ch/record/400313/files/p21.pdf. Luettu 10.3.2025.

2. COCO Dataset. *Common Objects in Context*. Verkkoaineisto: https://cocodataset.org/#home. Luettu 3.4.2025.

3. Core Electronics. *Getting Started with YOLO Object and Animal Recognition on the Raspberry Pi*. Verkkoaineisto: https://core-electronics.com.au/guides/raspberry-pi/getting-started-with-yolo-object-and-animal-recognition-on-the-raspberry-pi/. Luettu 29.3.2025.

4. GeeksforGeeks. *History of Python*. Verkkoaineisto: https://www.geeksforgeeks.org/history-of-python/. Luettu 26.3.2025.

5. Google Colab. *Google Colaboratory*. Verkkoaineisto: https://colab.research.google.com/github/EdjeElectronics/Train-and-Deploy-YOLO-Models/blob/main/Train_YOLO_Models.ipynb. Luettu 1.4.2025.

6. Google Colab. *Google Colab*. Verkkoaineisto: https://www.shiksha.com/online-courses/articles/how-to-use-google-colab-for-machine-learning-projects/ Luettu 1.4.2025.

7. Koodinkutoja. *Tekoälyn käyttäminen*. Verkkoaineisto: https://koodinkutoja.com/tekoalyn-kayttaminen/. Luettu 14.3.2025.

8. Label Studio. *Blogi ja opastus*. Verkkoaineisto: https://labelstud.io/blog/. Luettu 21.3.2025.

9. Mikrobitti. *Konenäön vallankumous on nurkan takana – käytetään jo Suomen teillä ja Senaatintorilla*. Verkkoaineisto: https://www.mikrobitti.fi/uutiset/konenaon-vallankumous-on-nurkan-takana-kaytetaan-jo-

suomen-teilla-ja-senaatintorilla/2064afd9-7792-4ac0-aef8-ade4363764fa. Luettu 14.3.2025.

10. OpenCV. *OpenCV – Open Source Computer Vision Library*. Verkkoaineisto: https://opencv.org/. Luettu 27.3.2025.

11. Portable As. *History of Machine Vision*. Verkkoaineisto: https://www.portableas.com/news/history-of-machine-vision/. Luettu 10.3.2025.

12. Python Software Foundation. *Python Classes and Object-Oriented Programming*. Verkkoaineisto: https://docs.python.org/3/tutorial/classes.html. Luettu 26.3.2025

13. Python Software Foundation. *Python Control Flow*. Verkkoaineisto: https://docs.python.org/3/tutorial/controlflow.html#defining-functions. Luettu 26.3.2025.

14. Raspberry Pi Foundation. *Raspberry Pi 4*. Verkkoaineisto: https://www.raspberrypi.org/. Luettu 25.3.2025.

15. Raspberry Pi Foundation. *Raspberry Pi Camera -moduuli*. Verkkoaineisto: https://www.raspberrypi.com/products/camera-module-v2/. Luettu 26.3.2025.

16. ResearchGate. *Industrial Application of Machine Vision*. Verkkoaineisto: https://www.researchgate.net/publication/271509417_INDUSTRIAL_APPLICATION_OF_MACHINE_VISION. Luettu 19.3.2025.

17. Szeliski, R. 2010. *Computer Vision: Algorithms and Applications*. Springer. Verkkoaineisto: http://szeliski.org/Book. Luettu 10.3.2025.

18. Szeliski, R. 2022. *Computer Vision: Algorithms and Applications (2nd ed.)*. Springer. Verkkoaineisto https://szeliski.org/Book/ Luettu 14.3.2025

19. Thonny. *Thonny Python IDE*. Verkkoaineisto: https://thonny.org/. Luettu 30.3.2025.

20. Ultralytics. *Machine Learning Models for Computer Vision*. Verkkoaineisto: https://www.ultralytics.com/. Luettu 20.3.2025.

Kuvalähteet

Kuva 1. Vision System Sorts Castings at General Motors Canada kirjasta https://link.springer.com/chapter/10.1007/978-3-662-09771-7_19

Kuva 2. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 3. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 4. Saarelainen, Elias 2025. Oma piirros.

Kuva 5. Saarelainen, Elias 2025. Oma valokuva.

Kuva 6. Partco, 2023. *Raspberry Pi 4B 4GB tuotesivu*. Verkkoaineisto: https://www.partco.fi/fi/raspberry-pi/raspberry-pi/20831-raspberry-pi4b-4gb.html.

Kuva 7. Raspberry Pi, 2023. *Camera Module V2 tuotesivu*. Verkkoaineisto: https://www.raspberrypi.com/products/camera-module-v2/

Kuva 8. Saarelainen, Elias 2025. Oma valokuva.

Kuva 9. Saarelainen, Elias 2025. Oma valokuva.

Kuva 10. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 11. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 12. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 13. Saarelainen, Elias 2025. Oma valokuva.

Kuva 14. Saarelainen, Elias 2025. Oma valokuva.

Kuva 15. Saarelainen, Elias 2025. Oma valokuva.

Kuva 16. Saarelainen, Elias 2025. Oma valokuva.

Kuva 17. Saarelainen, Elias 2025. Oma kuvakaappaus.

Kuva 18. The Pi Hut, 2023. *Dual Fan Heatsink for Raspberry Pi*. Verkkoaineisto: https://thepihut.com/products/dual-fan-heatsink-for-raspberry-pi