LAPIN AMK
Lapland University of Applied Sciences

SYNKRONOITU IOT-MONIKAMERAJÄRJESTELMÄ

Fish-IoT -hanke

Mikko Sippola

Opinnäytetyö

Tieto- ja viestintätekniikka
insinööri (AMK)

2022

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Mikko Sippola	Vuosi	2022
Ohjaaja	Anssi Ylinampa		
Toimeksiantaja	Lapin ammattikorkeakoulu		
Työn nimi	Synkronoitu IoT-monikamerajärjestelmä		
Sivumäärä	23		

Opinnäytetyön tavoitteena oli saada selville, kuinka saataisiin kuvattua tietty isompi alue tarkasti käyttäen useaa kameraa. Järjestelmä osaa yhdistää usean kameran ottaman kuvan yhdeksi isoksi kuvaksi. Lisäksi tavoitteena oli tutkia sitä, kuinka objektintunnistus saataisiin toimimaan käyttäen montaa kameraa yhtäaikaisesti.

Projektin rakentamisessa oli kaksi laitevaihtoehtoa, jotka olivat Rasberry Pi ja Jetson Nano. Päädyin ottamaan alustakseni Jetson Nanon.

Opinnäytetyön tilaajana toimi Älykäs rakennettu ympäristö -osaamisryhmän TKI ja hankkeena oli Fish-Iot-hanke, jonka tarkoituksena oli suunnitella Kalasydän Oy:lle automatisoitu konenäkökonsepti, jolla pystyttäisiin ottamaan automatisoidusti tarkkoja kuvia kaloista.

Projekti eteni alkuun ongelmitta. Jetson Nanon kokoonpano ja alustus oli alkuun hidasta, mutta ei kovin haastavaa. Sain opinnäytetyössäni aikaan monta erilaista tekoälytestiohjelmaa. Testiohjelmien avulla yritin selvittää tarkinta mahdollista objektintunnistustapaa tunnistamaan kaloja. Onnistuin tuottamaan tarkkoja tunnistustuloksia ohjelmilla. Ongelmat kuitenkin esiintyivät automaattisten kuvien ottamisessa. En onnistunut tekemään itselleni ohjelmaa, joka osaisi ottaa kuvan automaattisesti yhdellä tai useammalla kameralla tunnistaessaan halutun objektin.

Avainsanat Jetson Nano, konenäkö, objektintunnistus

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Study Programme in Information and
Communication Technology
Bachelor of Engineering

Author	Mikko Sippola	Year	2022
Supervisor	Anssi Ylinampa		
Commissioned by	Lapland University of Applied Sciences		
Subject of thesis	Synchronized IoT Multi-camera System		
Number of pages	23		

The aim of this thesis study was to find out how to accurately photograph a certain larger area using several cameras. The system would combine the image taken by several cameras into one large image. In addition, the goal was to investigate how object recognition could be made to work using several cameras at the same time. The commissioner of the study was Lapland UAS's R&D of the Intelligent Built Environment expertise group, and the project was the Fish-Iot project.

The purpose of the Fish-Iot project was to design an automated machine vision concept for Kalasydän Oy that would be able to automatically take accurate pictures of fish. Jetsen Nano was chosen as the platform to the project. At first, the project proceeded without any problems. The assembly and initialization of the Jetson Nano was slow, but not very challenging. Many different artificial intelligence test programs were created. With the help of the test programs, it was tried to find out the most accurate possible object recognition method to recognize fish.

However, problems occurred when taking automatic pictures. It turned out to be impossible to make a program that would automatically take a picture with one or more cameras when the desired object was recognized.

Key words Jetson Nano, object recognition, machine vision

SISÄLLYSLUETTELO

1 JOHDANTO

Objektitunnistuksen suosio on kasvanut valtavasti jokapäiväisessä käytössä ajan myötä. Objektin tunnistus on alkanut kulkemaan käsi kädessä myös tekoälyn kanssa. Erilaisten sulautettujen järjestelmien ja ohjelmien yleistymisestä jokapäiväisissä ympäristöissä on myös tullut jatkuvasti helpompaa ja suositumpaa, esimerkiksi tuotantolinjastoilla tuotteiden laadunvalvonnassa, tapahtumissa asiakasmäärien arvioinnissa, kameroiden automaatti tarkennuksessa tai liikenteen valvonnan työtehtävissä.

Ihmiskunnan teknologisen kehittymisen nopeuden ansioista neuroverkkojen kanssa työskentely ei ole enää vain yliopistojen laboratorioympäristöteknologiaa. Nykypäiväisen teknologiakehityksen ansioista kuka tahansa pystyy rakentamaan objektitunnistussovelluksen, tekoälyn tai automaattisen skriptin. Tilannetta helpottaa myös se, että verkko on täynnä avoimen lähdekoodin neuroverkkojen malleja, joista jokainen voi muovata haluamansa tai kehittää täysin uuden mallin juuri omiin tarpeisiin sopivaksi.

Tässä opinnäytetyössä on tavoitteena kehittää monikamerajärjestelmä käyttäen hyväksi Nvidia Jetson Nanon sulautettua järjestelmää, joka olisi kykenevä tunnistamaan objekteja. Osana lopullista tavoitetta olisi, että Nvidian Jetson Nano osaisi tunnistaa monen kameran avulla kaloja ja kalalajikkeita ja ottaa korkealaatuisen kuvan kalasta. Opinnäytetyö suoritettiin osana Fish-IoT hanketta yhteistyössä Älykäs rakennettu ympäristö TKI kanssa, joka toimi toimeksiantajana.

Aloitin opinnäytetyön selvittämällä, mille alustalle rakennuttaisin kamerajärjestelmän. Vaihtoehtoina olivat Rasberry Pi tai Jetson Nano. Päädyin valitsemaan Jetson Nanon, koska Jetson Nano on suunniteltu tekoälyläheiseksi järjestelmäksi ja on kykenevä neljän kameran samanaikaista käyttöön. Koska olin rakentamassa monikamerajärjestelmää tarvitsin keinon liittää Jetson Nanoon enemmän kameroita. Jetson Nanossa on kaksi MIPI CSI-2 porttia, joihin pystyy liittämään Rasberry Pi -kameran, joten tilasin koulun kautta Arducam Multi-Camera Adapterin, joka mahdollistaa neljän kameran liittämisen Jetson nanoon.

2 PROJEKTIN TYÖKALUT

2.1 Nvidia Jetson Nano “Developer Kit”

Nvidia Corporation on globaali teknologiayritys Yhdysvalloissa. Nvidia tunnetaan parhaiten näytönohjain (GPU) ja järjestelmäpiiri valmistajana. (Nvidia 2022a.)

Nvidia Jetson Nano (Kuvio 1) on nimensä mukaisesti pienikokoinen tietokone, joka on rakennettu erikoistumaan energiatehokkaaseen tekoälysovelluksien ylläpitämiseen. Jetson Nano on hintaansa ja kokoonsa nähden erittäin tehokas sulautettu järjestelmä. Jetson Nano sisältää ARM Cortex-A57 MP neliydin prosessorin ja Nvidian Maxwell 128 Nvidia CUDA® näytönohjaimen. Muistin puolesta Nvidia Jetson Nano sisältää 4GB 64-bit LPDDR4 -muistia 25.6GB/s nopeudella. Nvidia Jetson Nanossa on myös neljä USB 3.0 -porttia ja yksi micro-USB portti viiden voltin sisääntulolla, Ethernet portti, HDMI 2.0 ja DP 1.4 (DisplayPort) porteilla ja neljäkymmentäpinnisen GPIO portin. (Nvidia 2022b.)

Kuvio 1. NVIDIA Jetson Nano Developer Kit (Nvidia 2022c)

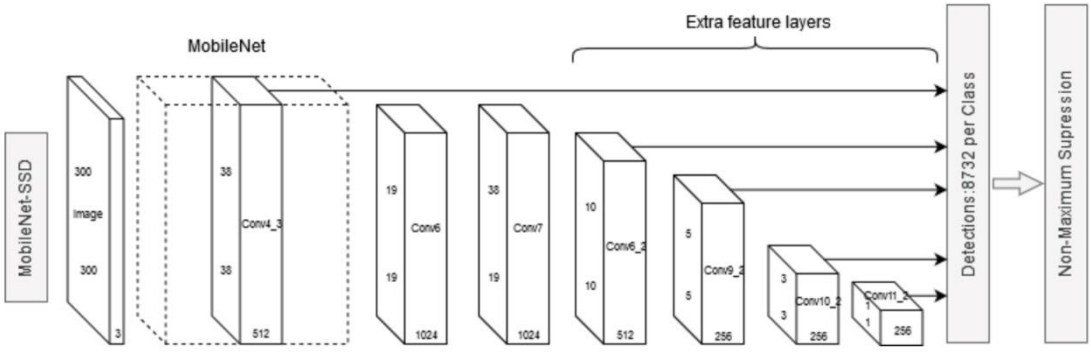
2.2 Python

Python on yksi yleisimpiä ohjelmointikieliä, jonka Guido van Rossum kehitti ja julkaisi helmikuun 20.2.1991. Python on yleinen ohjelmointikieli, joka tarkoittaa, että Pythonilla voidaan luoda erilaisiin tarkoituksiin sopivia ohjelmia. Python ohjelmointikieli ei erikoistu vain yhteen osa-alueeseen. (Python 2022a.)

Python on tulkittava kieli, joka tarkoittaa, että ohjelmat ovat valmiita ajettavaksi ilman että niitä tarvitsee kääntää ensiksi. Tämä lisää ohjelmien mahdollista testaamista lyhyessä ajassa. Pythonia voidaan käyttää moniin tarkoituksiin, kuten ohjelmistokehitykseen, verkkokehitykseen tai esimerkiksi komentosarjojen ohjelmointiin. (Python 2022b.)

2.3 Mobilenet-SSD

Mobilenet-SSD (Kuvio 2) on suosittu objektintunnistusmalli mobiili ja sulautetuille laitteille. Mobilenet-SSD käyttää Single-Shot multibox Detector(SSD) -neurovekkoa. Tämä tarkoittaa, että objektintunnistus ottaa yhden kuvan tai kehyksen kerrallaan ja tunnistaa kuvassa rajatun alueen. Syy miksi, SSD arkkitehtuuria käytetään paljon reaaliaikaisessa objektin tunnistuksessa, on sen nopeus suhteessa sen tarkkuuteen. Sillä ei ole delegoitua alueehdotusverkkoa sen sijaan SSD ennustaa raja-alueet ja luokat suoraan sille määritetyistä piirrekartoista. (Hui J. 2018.)



Kuvio 2. Mobilenet backbone (Franklin 2022)

Mobilenet itsessään käyttää virtaviivaistettua arkkitehtuuria. Mikä tarkoittaa, että Mobilenet käyttää syvyyskohtaista ja pistekohtaisia konvoluutioita. Mobilenet on

suunniteltu niin, että se rakennuttaa oman neuraaliverkonsa mahdollisimman kevyeksi, jotta laitteet, jotka sisältävät erittäin vähän laskentatehoa pystyvät käyttämään neuraaliverkkoa ongelmitta. (Sarkar 2021.)

2.4 OpenCV

OpenCV on Intelin rakentama avoimen lähdekoodin konenäkökirjasto. OpenCV on optimoitu tukemaan moniydinprosessoreita, joten OpenCV:tä pystytään käyttämään reaaliaikaisen kuvan käsittelyssä. OpenCV:n konenäkökirjastot tukevat useita käyttöjärjestelmäympäristöjä, kuten Android, iOS, Windows ja Linux. OpenCV on kirjoitettu C++ kielellä ja sitä voidaan käyttää useilla eri rajapinnoilla, joista yleisimpiä ovat C++, C, Java ja Python. OpenCV luotiin kiihdyttämään konenäkösovellusten kehitystä vuonna 1999 Intelin ansiosta. OpenCV toimii Convolutional Neural Networks (CNN) ja Deep Neural Networks (DNN) kanssa, jotta kehittäjät voivat rakentaa innovatiivisia ja tehokkaita uusia konenäkö ohjelmia. (Intel 2017.)

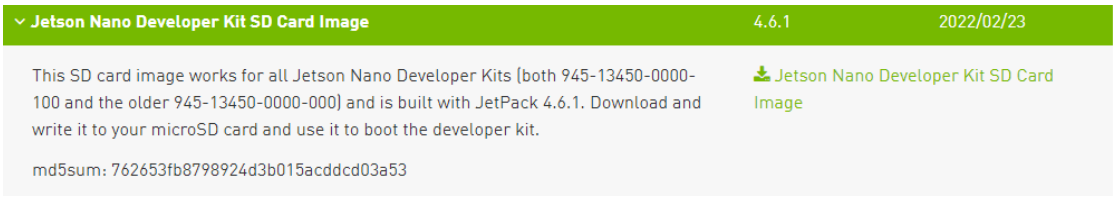
3 TOTEUTUS

3.1 Jetson Nanon asennus

Ennen Jetson Nanon varsinaista asennustyötä tarvittiin minimissään 16GB microSD-kortti, virtalähde (5V 2A) ja Ethernet-kaapeli tai WiFi-adapteri. Jetson Nanossa ei ole sisäänrakennettua WiFi-adapteria mutta on mainittava, että Jetson Nanoon on mahdollista asentaa WiFi-antennimoduuli, joka asennetaan jäähdytyssiilin alapuolelle.

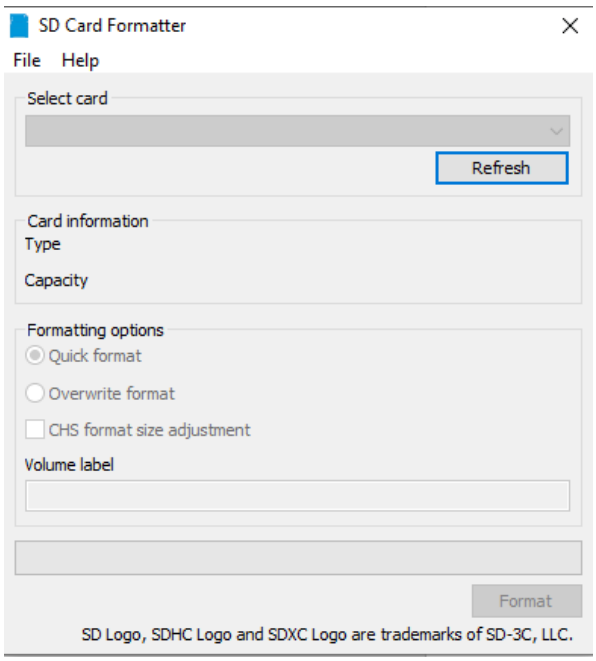
3.2 Ubundu käyttöjärjestelmän alustaminen muistikorille

Jetson Nanon itsessään ei sisällä käyttöjärjestelmää tai sisäistä kiinteää muistia. Käyttöjärjestelmä asennetaan muistikortille Jetson Nanon avulla ja kaikki järjestelmän tallennustila on kyseisellä muistikortilla. Alustamista varten ladattiin uusin versio Jetson Nano Developer Kit SD Card Image (Kuvio 3) tiedosto Nvidian developer sivustolta.



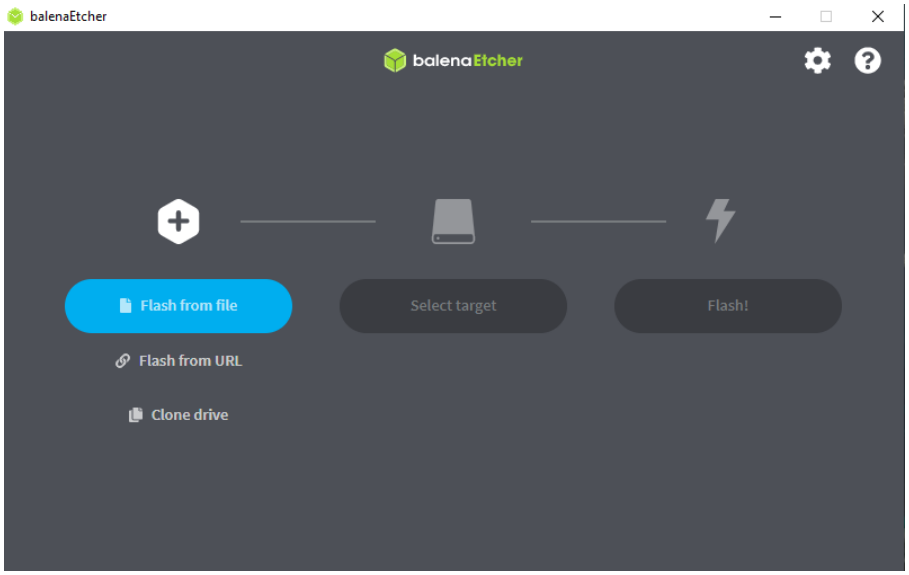
Kuvio 3. Jetson Nano Developer Kit SD Card Image (Nvidia 2022d)

Image itsessään pitää asentaa microSD muistikortille. Ennen image tiedoston asentamista muistikortti täytyy formatoida. Seurasin alustuksessa Nvidia ohjeistusta ja asensin SD Association valmistajan ohjelman nimeltä SD Card Formatter (Kuvio 4) muistikortin alustusta varten. Käytin hyväkseni SD muistikorteille USB-muistitikkua, johon oli mahdollista liittää microSD muistikortti, jonka avulla pystyin alustamaan microSD muistikortin.



Kuvio 4. SD Memory Card Formatter ohjelma

MicroSD muistikortin Formatoinnin jälkeen olin valmis alustamaan JetPack 4.6.1 version käyttäen apunani balena Etcher (Kuvio 5) ohjelmaa. Seuraavassa kuvissa 5 on havainnollistettu balenaEtcher ohjelman ulkoasua ja toimintoja.



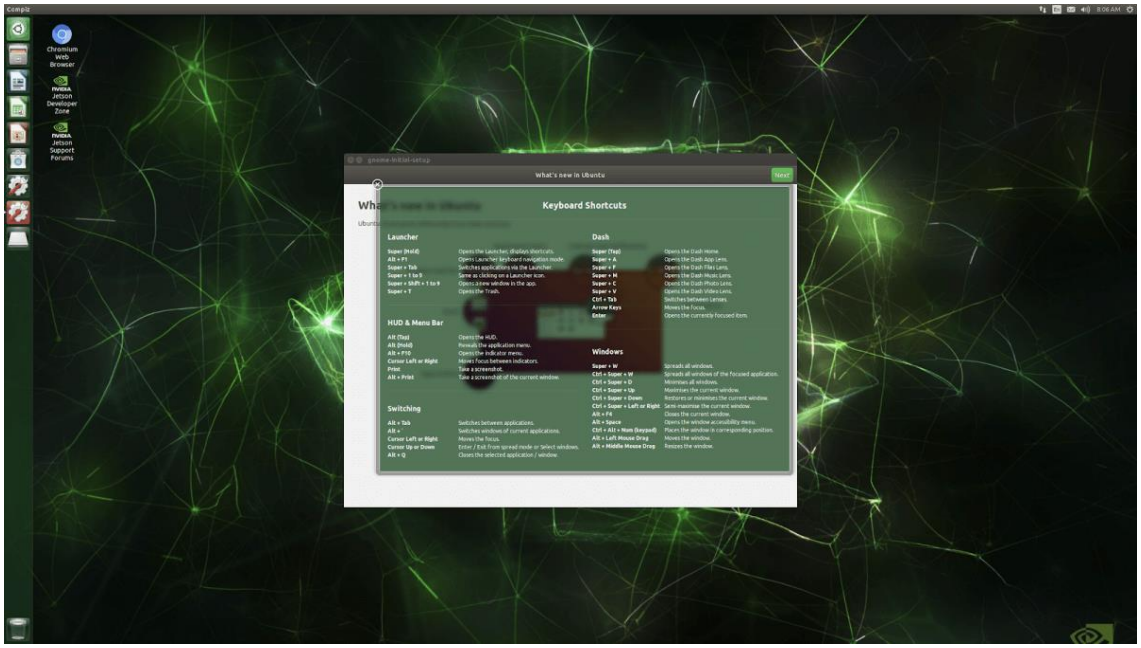
Kuvio 5. Balena Etcher ohjelma

Image-tiedoston alustamisen jälkeen syötin muistikortin Jetson Nanon microSD porttiin ja suoritin laitteen käynnistyksen ensimmäisen kerran. Ensimmäisellä käynnistys yrityksellä törmäsin kuitenkin ongelmaan. Jetson Nano ei pysty käyn-

nistämään itseänsä pelkän Micro-USB-liittimestä saadulla virralla toisin kuin Nvidia kehittäjä ohjeissa väitettiin. Jouduin siis etsimään itselleni käsiin uuden laturin, joka sopi Jetson Nanoon.

3.3 Jetson Nano Ubundu -käyttöjärjestelmän ja ohjelmistojen alustus

Ubundu käyttöjärjestelmän- ensimmäinen asennus sujui ongelmitta. Asennus itsessään oli erittäin yksinkertainen. Ubundu pyytää ensimmäisenä hyväksymään Nvidian käyttäjä lisenssisopimuksen tämän jälkeen päätettiin käyttöjärjestelmän kieli, käyttäjän sijainti ja tehdään käyttäjälle profiili.



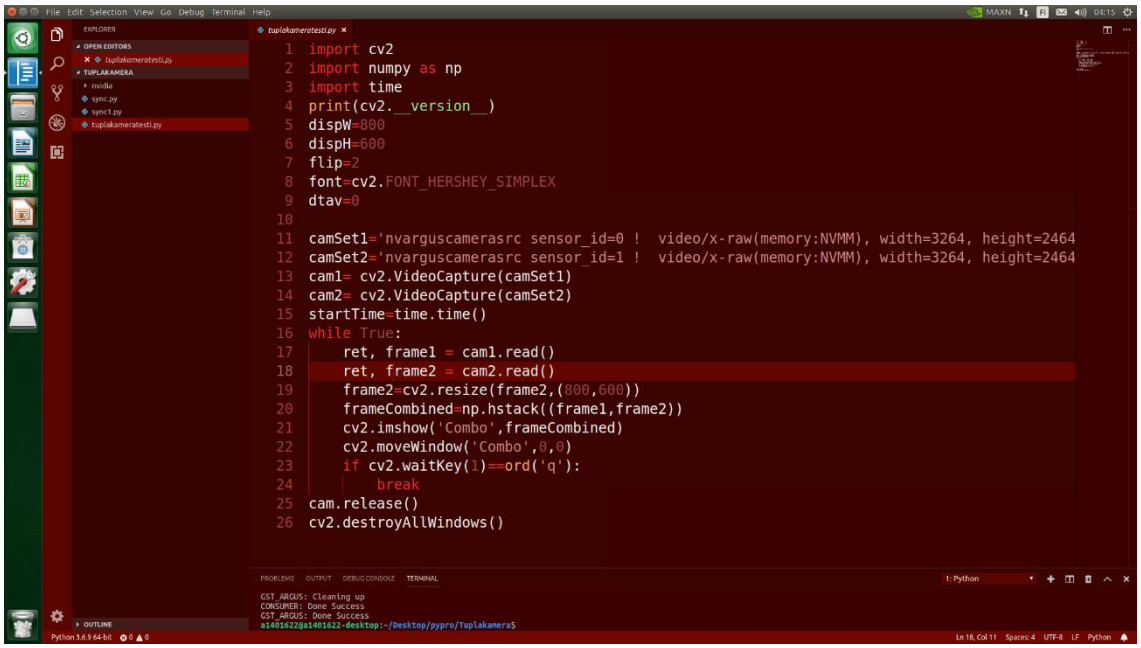
Kuvio 6. Ubundu 18.04 -työpöytä

Työpöydälle päästyäni etsin komentorivin avauskomennon ja suoritin käyttöjärjestelmän päivityskomennon ”*sudo apt-get update*” joka etsii viimeisimmät päivitykset tietolähteestä. Käyttöjärjestelmänpäivityksen jälkeen halusin testata Pythonin toimivuutta, joten tein nopean ”*hello world*” koodin ja asensin testinä Nano teksitieditorin. Ensimmäisenä asensin NumPy-kirjaston. NumPy on Pythonille tehty ohjelmointikielen kirjasto, joka on tarkoitettu taulukoiden kanssa työskentelemiseen. Numpy sisältää myös toimintoja lineaarialgebraa, fourier-muunnosta ja matriisien kanssa työskentelemistä varten. NumPy on avoimen lähdekoodin projekti, jota voi käyttää ja muokata vapaasti. NumPy tulee lyhenteestä Numerical

Python. (W3Schools 2022.) Seuraavaksi asensin Matplotlib-kirjaston, joka on laajennus NumPy-kirjastoon. Matplotlib tarjoaa oliopohjaisen ohjelmointirajapinnan yhteistyössä NumPy-kanssa. Tämän avulla Python voi luoda staattisia, animoituja tai interaktiivisia visualisointeja toteutettavasta koodista. (Matplotlib 2022.)

3.4 Valmiin koodin kokeilu

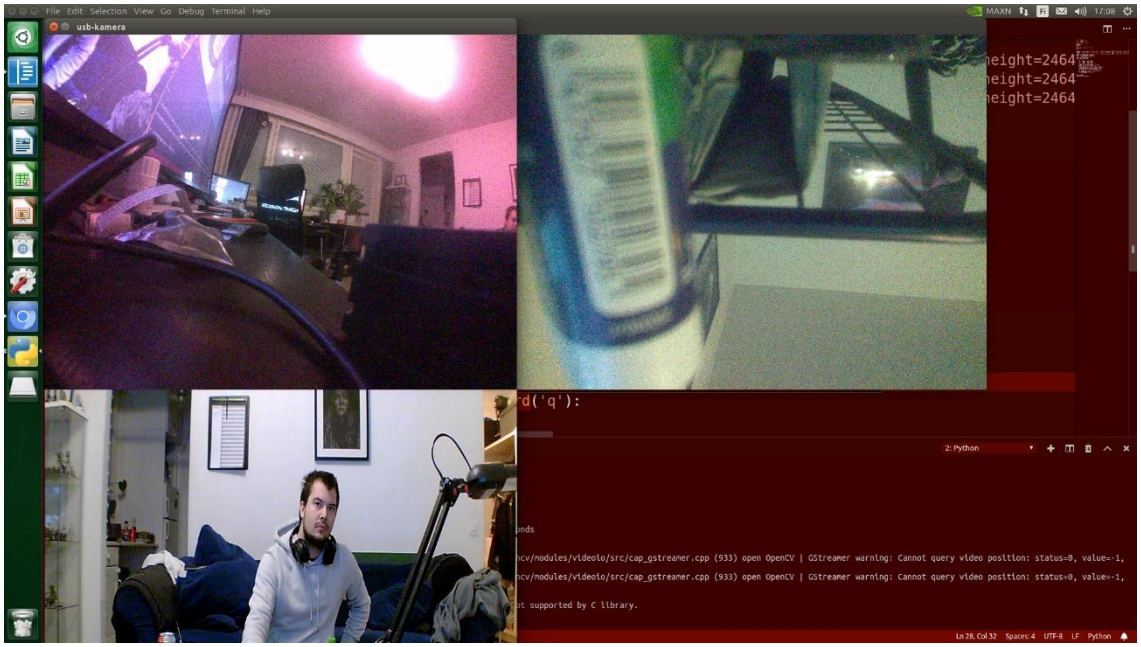
Pythonin ympäristön alustettuani päätin lähteä kokeilemaan valmiiksi kirjoitettuja testi ohjelmia samalla opiskellen Ubundu-käyttöjärjestelmän komentoja. Ensimmäiseksi tavoitteekseni otin Rasberry Pi -kameroiden testaamisen käyttäen Pythonia. Löysin tarkan opastusvideon, jonka avulla pystyin kirjoittamaan hyvin yksinkertaisen koodin kahden samanaikaisen kameran testaukseen tehden pieniä muutoksia alkuperäiseen koodiin. Tämä testaus mahdollisti myös pienen perehtymisen OpenCV:n käyttämiseen.



Kuvio 7. Kahden RPI kameran samanaikainen käyttö

Jatkoin koodia hieman ja pienten lisämuokkauksien jälkeen sain Pythonin toistamaan kolmea kameraa yhtäaikaisesti. Jouduin lisäämään Jetson Nanoon yhden

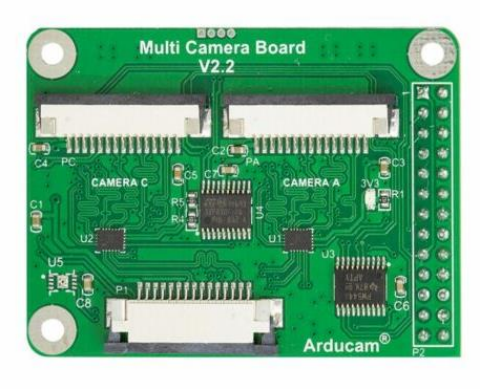
USB-kameran mahdollistaakseni lopputuloksen. Mahdollisuus neljännelle kameralle olisi ollut koodin puolesta, mutta valitettavasti itse kamerat loppuivat minulta kesken.



Kuvio 8. Kolmen samanaikaisen kameran kuvan toistaminen OpenCV:n avulla

Alkuperäisenä suunnitelmana oli saada neljä yhtäaikaista kameraa toimimaan Jetson Nanon kanssa. Tätä tavoitetta varten olin tilannut Lapin ammattikorkeakoululta Jetson Nanoon sopivan adapterin monen kameran yhtäaikaista ajamista varten. Laitteena toimi Arducamin monikamera-adapteri versio 2.2.

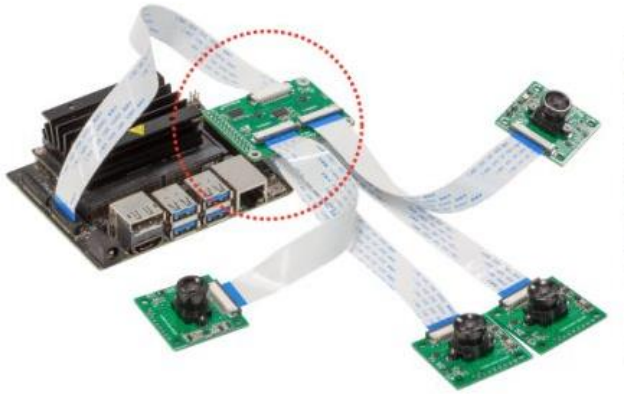
Arducam multi-camera adapter (Kuvio 8) on alunperin suunniteltu toimimaan Raspberry Pi -malleilla A/B/B+, Pi 2 ja Raspberry Pi 4, 3,3b+. Adapteri tukee 5MP OV5647-, 8MP IMX219- ja 12MP IMX477 mallien kameroita ja erilaisten kameroiden sekoittamista ei suositella. GPIO-pinnien avaaminen on vaadittavaa laitteen toimivuuden takaamiseksi. (Arducam 2022a.)



Kuvio 8. Arducam Multi Camera Adapter Module V2.2 for Raspberry Pi (Arducam 2022a)

Arducamin monikamera-adapteri (Kuvio 9) osoittautuikin paljon haastavammaksi laitteeksi kuin olin aluksi osannut odottaa. Adapteri on alun perin suunniteltu Rasberry Pi -alustalle, joten Jetson Nanon ohjeet ja testausympäristökoodit olivat vähäiset.

Sain adapterin kuitenkin toimimaan, mutta valitettavasti en tavalla mikä oli tavoite. Sain kaikki neljä kameraa käymään päällä vuorotellen ja tallentamaan kuvankaappauksen jokaisesta kamerasta muistikorille. Tavoitteeni oli kuitenkin alunperin saada adapteri näyttämään kuvaa kaikista neljästä kamerasta samaan aikaan. Tämä ei kuitenkaan koskaan toteutunut. Pitkän tutkimustyön jälkeen sain selville, että kyseinen adapteri ei pysty välittämään kuvadataa kuin yhdestä kamerasta kerrallaan. Näin ollen jouduin muuttamaan keinoa, jolla yhdistän neljän kameraa samanaikaisesti Jetson Nanoon. (Arducam2022b.)

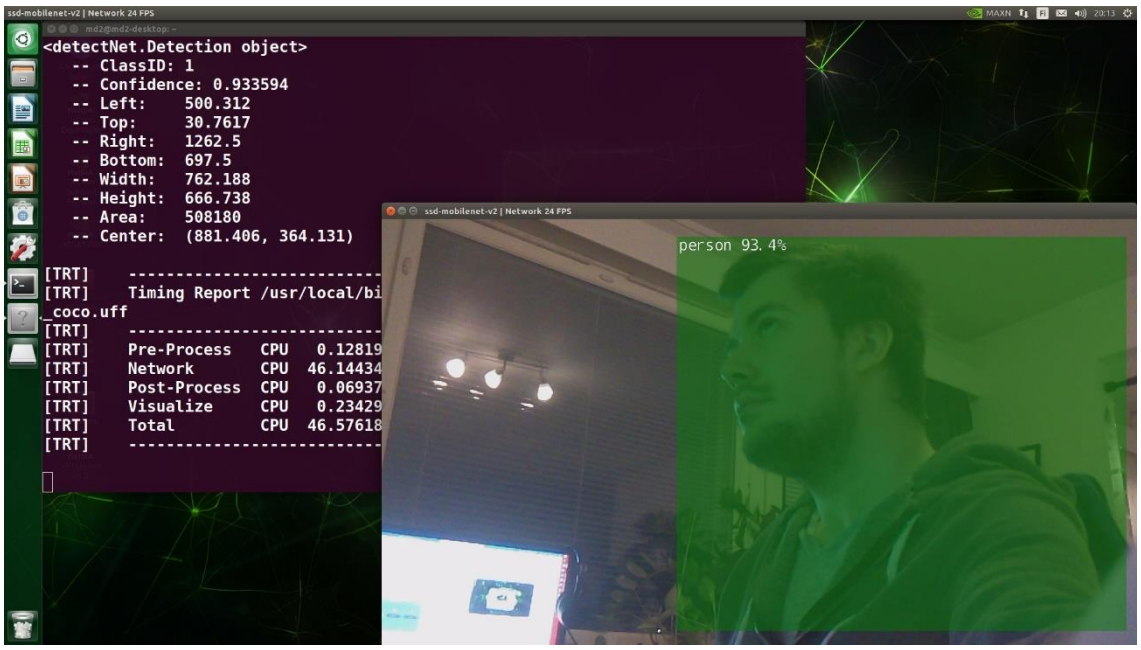


Kuvio 9. Alkuperäinen tavoite Jetson Nano Arducam adapteri asennus neljällä kameralla (Arducam 2022c)

Yhtenä hankkeen alkuperäisistä tavoitteista oli saada Jetson Nano ottamaan monen kameran avulla kuvat, joka yhdistettäisiin yhdeksi isoksi kuvaksi. Kuvien yhdistäminen ei kuitenkaan onnistunut niin helposti kuin oletin. Ideani oli tehdä koodi, joka tallentaisi videosta kuvan kaappauksen Jetson Nanon muistiin nappia painamalla. Tämä kuitenkin muodostui hankalammaksi kuin oletin oman tietotaitoni takia ja jouduin luopumaan idean toteuttamisesta.

3.5 Konenäkö

Viime vuosien aikana keinotekoisen älykkyyden, neuroverkkojen ja syväoppimisen kehittyminen on ollut erittäin nopeaa, joka on mahdollistanut kuluttajille helpon pääsyn sisään konenäön kehittämiseen. Kuluttajien helppo aloitteisuuden ansioista internet sisältää eri alustoille tehtyjä avoimen lähdekoodin ohjelmia. Yhtenä opinnäytetyön tavoitteistani oli rakentaa konenäköön perustuva kamerajärjestelmä, joka osaisi tunnistaa sille opetettuja objekteja. Lähdin tutkimaan Nvidian kehittäjänopaan kurssia Jetson AI Fundamentals, jossa käytiin läpi reaaliaikaista objektin tunnistusta. Latasin itselleni Nvidian kehittäjä GitHub-tietolähteen, jonka avulla sain pääsyn testaamaan TensorRT-rajapintaa. TensorRT on Nvidian ylläpitämä machine learning framework joka on optimoitu toimimaan parhaiten Nvidian sulautetuilla järjestelmillä. Nvidian ohjeet objektintunnistukseen kameran avulla olivat selkeät, mutta toteutus itsessään oli erittäin aikaa vievä. Nvidian ohjeissa käytettiin detectNet objektin tunnistusta SSD-Mobilenet-V2:sen kanssa. DetectNetin avulla tekoäly osaa rajata ja tunnistaa kuvassa olevat objektit, jotka löytyvät SSD-Mobilenet-V2 objektin tunnistus kirjastosta. Testikoodi toimi pienten koodimuutosten jälkeen. Koodi toimi tarkoituksen mukaisesti ja tuotti halutun lopputuloksen. Kuviossa 10 havainnollistan testikoodin ohjelmaa toiminnassa.



Kuvio 10. Yksinkertainen objektitunnistus DetectNetin ja SSD-Mobilenet-V2:sen avulla

DetectNet yhteistyössä SSD-Mobilenet-V2 mallikirjaston kanssa tekoäly osaa luokitella objekteja laajasti, mutta ei selvästikään tarkasti. Hetken testauksien jälkeen lähdin kokeilemaan hieman erilaista objektintunnistus mallia. Päätin kokeilla erilaista tunnistuskirjastoa ja tunnistusmetodia. Löysin ohjeet, joiden avulla pystyin kokeilemaan ImageNet nimistä laajaa datakeskusta Googlen GoogleNet kuvantunnistus mallin kanssa. ImageNet toimii yhteistyössä GoogLeNet-kuvantunnistusmallin kanssa. (Kuvio 11)

Kuvio 11. ImageNet ja GoogLenet avulla toteutettu kuvantunnistus

Testasin kuvakirjaston laajuutta ja tarkkuutta. Esitin puhelimeni kautta selviä kuvia kaloista. Osan kuvista tekoäly osasi tunnistaa vaikkakin hieman oudoilla termeillä. Kun näytin esimerkiksi kuvan hauesta puhelimestani tekoäly antoi tekstin ”barracouta, snoek”. Nämä ovat osittain oikein koska snoek on hollanniksi hauki ja barracouta on myös hauen lajike. GoogleNetin ja ImageNetin yhdistäminen vaikutti lyhyen kokeiluajan aikana varteenotettavalta tunnistustavalta. Näitä kahta voitaisiin käyttää yhteistyössä tarvittaessa tulevaisuudessakin, mikäli käytetään valmiita kuvantunnistuskirjastoja. DetectNet ja SSD-Mobilenet kokeiluissa tekoäly osasi rajata kuvan hauesta, mutta vain sanalla, fish joten ImageNetin ja Googlenetin on kykenevä havaitsemaan eri lajikkeita, joka oli yksi alkuperäisen Fish-IoT projektin tavoitteista.

3.6 Tekoälyn itseopetus

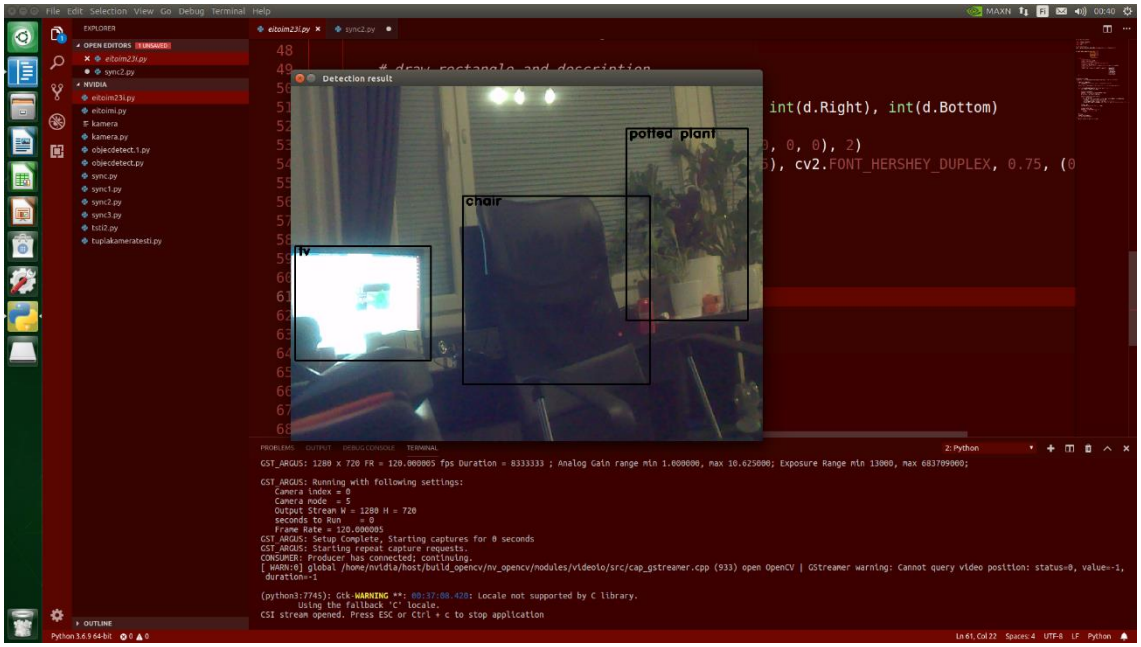
Fish-IoT projektin tavoitteena oli luoda tekoäly, joka olisi kykenevä tunnistamaan kaloja. Päätin kokeilla tekoälyn itse kouluttamista. ImageNetin ja GoogLeNetin yhteistyö oli kykenevä eri lajien tunnistukseen, halusin rajata tunnistusta itse kouluttamalla tekoälyn. Joten otin seuraavaksi tavoitteekseni yrittää kouluttaa itse tekoälyä tunnistamaan objekteja sille koulutetuilla parametreillä.
Oman tekoälyn kouluttamiseen löytyy myös ohjeita joidenka avulla voi aloittaa helposti oman tekoälyn kouluttamisen. Lähdin kokeilemaan tekoälyn koulutuksen testaamista käyttäen apuna Hello AI Wordl Nvidian Re-training SSD-Mobilenet kurssia. Koulutus toimii syöttämällä tietoverkolle suuren määrän rajattuja kuvia objekteista tai objektista, jota halutaan, että tietoverkko oppisi tunnistamaan. (Franklin 2022.)

Tekoälyn itseopetus tekniikat ovat kuittenkin erittäin aikaa vieviä ja vaativat satoja tai jopa tuhansia kuvia, jotta objektintunnistus toimisi kunnolla. Sen sijaan että ottetaan itse kuvia samasta objektiivista voidaan käyttää apuna valmiita kuvakirjastoja kuten Nvidian ohjeissa käytettiin. Valmiina kuvakirjastona toimi Google

Open Omages Dataset V7 joka sisältää yli 600 erilaista objektityyppiä. Näitä erilaisia objektityyppejä hyväksikäyttäen pystytään luomaan tarkka kuvan tunnistus tekoäly juuri omiin tarkoituksiin. Kouluttaa tekoälyä sitten valmiita kirjastoa käyttäen tai itse, prosessi on aina erittäin aikaa vievää. Kokemattomuuden ja tiukan aikatauluni takia jouduin valitettavasti jättämään tekoälyni itseopetuksen vain teoria tasolle ja siirtymään takaisin käyttämään valmiita objektin tunnistusarkkitehtuureita kuten DetectNet:iä.

3.7 Objektintunnistus OpenCV:n ja Mobilenet-SSD:n yhteistyössä

Viimeisessä kokeilussani päätin testata kuinka OpenCV:n ja Mobilenetin objektintunnistus toimivat yhteen. Pienen tutkimisen jälkeen löysin muutaman selvän esimerkin, joiden pohjalta pystyin lähteä kokeilemaan objektintunnistuksen toimivuutta helposti. Muutaman erilaisen esimerkkikoodin testaamisen jälkeen löysin mallikoodin, jota osasin lukea ja ymmärtää. Pienten testauksien ja muutoksien jälkeen koodi tuotti kuvan (Kuvio 12), jota halusin käyttää hyödyksi tulevaisuudessa.



Kuvio 12. Objektintunnistus OpenCV ja Mobilenet-SSD avulla

Objektintunnistusohjelman toimimaan saannin jälkeen halusin yhdistää koodin käyttämään kolmea kameraa samaan aikaan vain yhden sijasta. Tässäkohtaa

minulla tuli kuitenkin seinä vastaan. Ongelmaksi muodostui GStreamer pipelinen käyttäminen suoraan koodista. GStreamer pipeline -koodin käyttö luonnistui yhdellä kameralla, mutta parametrien määritys, koodin muokkaus tukemaan monikamerajärjestelmääni ja USB- kameran luokittelu osoittautuivat tässä kohdassa omien tietotaitojeni ulkopuolelle.

4 POHDINTA

Yksinkertaisen konenäköohjelman tekeminen ei aloittelevalle ohjelmoijalle tuntunut ylitsepääsemättömän haastavalta. Ohjeet olivat yleensä helpposelitteisiä ja hahmotettavia elementtejä ei ollut samaan aikaan useita, joita olisi pitänyt ymmärtää. Kameroiden testaaminen oli alkuun helppoa selkeiden ohjeiden ja OpenCV:n yksinkertaisten komentojen ansioista.

Isoin ongelma oli itse alkaa selvittämään ratkaisua monikamerajärjestelmäkonseptini toteuttamiseen. Ilman valmista tietämystä konenäköympäristöstä ja vain pienellä kokemuksella Ubuntu -ympäristöstä testaaminen oli alkuun hidasta ja tuskallista.

Valmiita ohjeita ei myöskään suoraan löytynyt tai esimerkkitoteutuksia, jotka olisivat olleet lähellä haluamaani lopputulosta. Jouduin useasti rakentamaan ja kokeilemaan erilaisia vaihtoehtoja, joista yksikään ei tuottanut haluttua tulosta. Isoimmaksi vaivaksi kuitenkin muodostui teknillisen sanaston lukeminen. Useasti luulin löytäneeni ongelmaani vastauksen. Jotta sain kuitenkin selvää vastauksesta, jouduin käymään läpi paljon konenäön teknillistä sanastoa, johon oma ymmärrykseni oli erittäin pieni. Sanaston tulkinnan jälkeen sain useasti selville, että ongelma, johon luulin löytäneeni vastauksen, ei ollutkaan samanlainen ongelma kuin itselläni. Kuvakirjastojen opettaminen ja niiden itse tekeminen oli mielenkiintoista oppia. Tekoälyn opettaminen itsessään kuvakirjastojen avulla on aikaa vievää. Tekoälyn itseopetus ei vaikuttanut myöskään mahdottomalle täysin aloittelevalle tekijälle. Suurin ongelma on varmasti aika. Kuvakirjastojen opettaminen on aikaa vievää. Jotta tekoäly tunnistaisi myös asioita, joita ei löydy opetuskirjastoista täytyy ne opettaa itse. Kuvien ottaminen ja itse rajaaminen on aikaa vievää, tähän päälle vielä niiden opettaminen tekoälylle.

Monen kameran liittäminen järjestelmään osoittautui helpoksi, kunhan käytti USB-kameroita. Ongelmat tulivat vastaan, jos käytössä oli vain Rasberry Pi -kameroita kameraporttien puutteen vuoksi. Yritin kiertää tämän ongelman adapte-

riratkaisuni avulla, mutta valittavasti se ei onnistunut. Ilmeisesti adapteri, jolla ongelmani voitaisiin korjata, on olemassa, mutta minulla ei ollut aikaa tutustunut vaihtoehtoisiinadaptereihin.

Ammattisanasto ja sekavan termistön ansioista en useasti voinut käyttää hyödyksi mitään löytämiäni vastauksia, vaikka niistä olisi ehkä ollut hyötyä. Suunnitelmani oli ottaa usean kameran GStreamer pipeline -koodi ja muokata se yhdeksi GStreamer pipeline -koodiksi. Tutkimuksieni pohjalta on GStreamer pipelinien yhdistäminen pitäisi olla mahdollista, mutta kaikki kokeiluni epäonnistuivat. Toinen vaihtoehto olisi ollut saada sama ohjelma lukemaan useaa GStreamer-koodia eri instansseissa. Koetin löytää myös vastausta tällaiseen ratkaisuuni, mutta en löytänyt mitään vastausta, mikä olisi vastannut kysymystäni.

Uskon että, jos minulla olisi ollut enemmän aikaa ja resursseja, olisin pystynyt toteuttamaan haluamani lopputuloksen. Tässä vaiheessa jouduin kuitenkin lopettamaan ohjelman kehittämisen, koska opinnäytetyölle varattu aika oli loppumassa.

Kaiken kaikkiaan opinnäytetyöprojekti on ollut minulle opettavainen. Kun aloitin, minulla ei ollut juuri minkäännäköistä kokemusta Jeton Nanon tai Rasberry Pin kaltaisista sulautetuista järjestelmistä. Ohjelmointitaitoni ovat myös parantuneet huomattavasti työskennellessäni näin haastavan projektin parissa.

LÄHTEET

Arducam 2022a. Arducam Multi Camera Adapter Module V2.2 for Raspberry Pi 4 B, 3B+, Pi 3, Pi 2, Model A/B/B+, Work with 5MP OV5647 / 8MP IMX219 / 12MP IMX477 Cameras. Viitattu 1.12.2022 https://www.arducam.com/product/multi-camera-v2-1-adapter-raspberry-pi/?utm_source=youtube-vid&utm_medium=viddescription.

Arducamb 2022b. Arducam Multi-Camera Adapter on the Jetson Nano & Xavier NX. Viitattu 22.11.2022 https://www.arducam.com/docs/camera-for-jetson-nano/multiple-cameras-on-the-jetson/arducam-multi-camera-adapter-on-the-nano-xavier-nx/.

Arducam 2022c. Arducam Multi-Camera Adapter on the Jetson Nano & Xavier NX. Viitattu 22.11.2022 https://www.arducam.com/docs/camera-for-jetson-nano/multiple-cameras-on-the-jetson/arducam-multi-camera-adapter-on-the-nano-xavier-nx/.

Frankin 2022. Re-training SSD-Moblinet. Nvidia Jeton-inference. Viitattu 23.11.2022 https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-ssd.md.

Hui J. 2018. SSD object detection: Single Shot MultiBox Detector for real-time processing. Viitattu 24.11.2022 https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.

Intel 2017. What is OpenCV? Viitattu 27.11.2021 https://www.intel.com/content/www/us/en/developer/articles/technical/what-is-opencv.html

Matplotlib. Matplotlib: Visualization with Python. Viitattu 19.11.2022 https://matplotlib.org/.

Nvidia 2022a. Nvidia About Us. Viitattu 14.11.2022 https://www.nvidia.com/en-us/about-nvidia/#slide-14-095fee10.

Nvidia 2022b. Technical Specifications Nvidia Developer. Viitattu 14.11.2022 https://developer.nvidia.com/embedded/jetson-nano.

Nvidia 2022c. Jetson Nano Developer Kit. Viitattu 7.1.2023 https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

Nvidia 2022d. Jetson Download Center. Viitattu 14.11.2022 https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano

Python 2022a. General Information. Viitattu 13.11.2022 https://docs.python.org/3/faq/general.html#what-is-python.

Python 2022b. General Information. Viitattu 13.11.2022 https://docs.python.org/3/faq/general.html#what-is-python.

Sarkar 2021. Understanding Depthwise Separable Convolutions and the efficiency of MobileNets. Viitattu 24.11.2022 https://towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficiency-of-mobilenets-6de3d6b62503.

W3Schools 2022. NumPy Introduction. Viitattu 19.11.2022 https://www.w3schools.com/python/numpy/numpy_intro.asp.