Tampereen ammattikorkeakoulu

Konenäön hyödyntäminen kaupan alan markkinoinnissa

Miiro Opas

Pasi Rintamäki

OPINNÄYTETYÖ
Toukokuu 2019

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

OPAS, MIIRO & RINTAMÄKI, PASI:
Konenäön hyödyntäminen kaupan alan markkinoinnissa

Opinnäytetyö 41 sivua, joista liitteitä 2 sivua
Toukokuu 2019

Opinnäytetyön tarkoituksena oli vertailla muutamaa erilaista konenäön menetelmää ja pohtia niiden käyttöä kaupan alan markkinoinnissa tarkoituksena evaluoida mahdollisuutta luoda tuote tai palvelu, jota tarjota kaupan alan yrityksille.

Alun teoria osuudessa paneuduttiin kone- ja tietokonenäön hyödyntämiseen kaupan alalla, kuten mihin kysymyksiin voidaan vastata hyödyntämällä kyseisiä tekniikoita. Samalla tarkasteltiin haasteita mitä saattaa nousta vastaan, kuten valvontakameroiden huonoon laatuun ja niiden oudot kuvakulmat.

Kehitysympäristön toteuttamiseen käytettiin palvelinta, johon oli asennettu Unix-pohjainen Linux jakelu. Palvelimelle asennettiin koneoppimiseen, konenäköön ja tietokonenäköön tarkoitetut työkalut ja niihin sidonnaiset ohjelmakirjastot. Käytettävän järjestelmäkokoonpanon dokumentaatio oli kattava, jonka pohjalta palveluiden ja työkalujen käyttöönotto on yksinkertaista.

Työssä käytiin läpi kaksi konenäön menetelmää. Nämä olivat tiheyskartan luonti ihmisten liikkeistä, sekä ihmislaskuri, jolla pystyy seuraamaan ihmisten liikettä kaupassa ja analysoimaan heidän kaupassa käyttämäänsä aikaa.

Tuloksien perusteella voidaan todeta, että on mahdollista luoda tuote tai palvelu, joka voisi hyödyntää kone- ja tietokonenäkö tukemaan kaupan alan markkinointia ja muuta toimintaa käyttäen työssä evaluoituja menetelmiä.

Varsinaisen tutkimustyön lisäksi työn tarkoituksena oli luoda kirjallinen dokumentaatio mahdollista jatkokehittämistä varten, jolla asiaan kiinnostuneet henkilöt voivat itse luoda oman kopion käytetystä kehitysympäristöstä.

Toimeksiantaja työlle oli Floating Point Society Oy.

---

konenäkö, tietokonenäkö, markkinointi, tieteellinen tietojenkäsittely

ABSTRACT

Tampere University of Applied Sciences
ICT Engineering
Software Engineering

OPAS, MIIRO & RINTAMÄKI, PASI:
Utilizing machine vision in commercial sector marketing

Bachelor's thesis 41 pages, appendices 2 pages
May 2019

The purpose of the thesis was to compare a few different machine vision methods and consider the possible use-cases in the commercial sector marketing and the possibility of creating a product or a service to trade sector companies.

The theory focuses on the use of machine and computer vision in the field of commerce, such as what questions can be answered by the techniques that were utilized, as well as the challenges that might arise, such as the poor quality of surveillance cameras and odd or inefficient angles.

A server with a Unix-based Linux distribution was used to implement the development environment that was used during the project. The server was equipped with the tools for machine learning machine vision and computer vision, and related program libraries that were needed for the system to work correctly. The documentation of the system and its configuration was relatively easy to comprehend, thanks to the ease-of-use documentation provided by their respective communities, making it simple to deploy the services and tools.

Two machine vision methods were used in the thesis. These were the creation of a density map that consisted of human movement, as well as a human counter to monitor the movement of people in or out of a given area, such as the interior of a shop, to determine the time spent on shopping.

Based on the results, it is possible to create a working solution, product or a service that could utilize the two computer vision methods to support the marketing and trade sector activities to give a better sense of the human behavior involved in the companies' respective field of trade.

In addition to the actual research, the secondary purpose of the work was to provide documentation for the possibility of further development, whereby interested individuals can create their own exact copy of the used development environment, hence replicating the results.

The client for the thesis was Floating Point Society Ltd.

Key words: machine vision, computer vision, marketing, scientific data processing

SISÄLLYS

ERITYISSANASTO

Konenäkö	Tieteen ala, joka kuvaa tietokoneiden matalatasoista kuvien ja videoiden ymmärtämistä.
Tietokonenäkö	Tieteen alla, joka kuvaa tietokoneiden korkeatasoista kuvien ja videoiden ymmärtämistä.
Konsulentti	Organisaation ulkopuolinen neuvoja tai asiantuntija. Kaupan alalla yrityksen ulkopuolinen tuote-esittelijä.
Python	Korkean tason ajoaikana tulkattava ohjelmisto ja skripti kieli.
Frame skipping	Kehyksen ohitus, ohjelma ohittaa tiettyjen kehysten (en, frame) näyttämisen tasoittaakseen animaation toisto nopeutta visuaalisen laadun kustannuksella.
Kehys	Videon yksi kuva (en, frame).
Bash	GNU-projektin luoma Turing-täydellinen komentotulkki.
Sudo	Komento, jota käyttämällä etuliitteenä ennen muita komentoja saadaan pääkäyttäjän oikeudet Unix-pohjaisissa käyttöjärjestelmissä.
Anaconda	Ilmainen avoimen lähdekoodin tieteellinen Python-distribuutio.
Conda	Anacondan sisältämä paketinhallintajärjestelmä Python ohjelmointikielelle.
Cmake	Avoimen lähdekoodin ohjelmisto, jolla voidaan tuottaa ohjaustiedostot järjestelmän omalle kääntäjälle omien komentotiedostojen avulla.

Make	Unix-järjestelmissä yleisesti käytetty työkalu, jolla voidaan koota lopullinen ohjelma useista lähdekoodeista käännettävistä objektitiedostoista.
Jupyter	Avoimen lähdekoodin web-applikaatio, jolla voidaan esitellä ja ajaa Python-lähdekoodeja, sekä esittää tieteellisiä tekstejä visuaalisemmin ja koota niistä kuvaajia.
Wrapper	Yleisesti viitataan ohjelmoinnissa suodattimeen (tai välikerros tulkkiin), joka käytännössä muuntaa tai ”käärii” eri ohjelmointikielellä toteutetut funktiot ja/tai metodit yhtenäisellä tavalla jonkin toisen ohjelmointikielen käytettäväksi.

1 JOHDANTO

Tutkimuskohteena oli evaluoida muutama kone- ja tietokonenäön menetelmä ja tutkia niiden hyödyntämistä kaupan alan markkinoinnin tukena. Tutkittaviksi menetelmiksi valittiin asiakkaan kanssa tiheyskartoitus, sekä ihmisten tunnistus ja laskeminen.

Tutkimuksen tarkoituksena oli kartoittaa mahdollisuuksia kehittää tuote tai palvelu yrityksille, joka perustuisi kaupan alan asiakkaiden liikkumisen analysointiin.

Kone- ja tietokonenäön käyttämiselle löytyy useita erilaisia toteutuksia ja tässä työssä niistä käytiin läpi kaksi, joista kumpikin käyttivät pohjanaan OpenCV:n tietokonenäkökirjastoa. Ensimmäinen toteutus käytti OpenCV:n threshold metodia luodakseen tiheyskartoituksen videolla näkyvästä liikkeestä ja toinen toteutus käytti OpenCV:tä yhdessä Dlib:in ja Caffe:n kanssa tunnistaakseen ihmisiä neuroverkolla ja laskeakseen heitä.

Toteutusten ajoa varten luotiin palvelinympäristö, jolla töitä pystyttäisiin tekemään etäyhteyden kautta riippumatta sijainnista. Palvelin ympäristö mahdollisti myös nopeammat ajoajat.

Toteutuksissa käytettiin testaamiseen Alon Lernerin julkaisemaa videota University Students (students003.avi). Kyseinen video on saatavilla UCY Computer Graphics Lab:in sivuilta. (Alon Learner)

2 KONE- JA TIETOKONENÄKÖ

Automated Imaging Associationin (AIA) mukaan konenäkö pitää sisällään kaikki teolliset ja ei-teolliset toteutukset, joissa käytetään kuvan tai videon käsittelyyn automatisoitua ohjelmistoa datan erottamiseen kuvasta. Konenäkö eroaa automaattisesta kuvan käsittelystä sillä, että ulostulo on kuvan sijaan dataa. Konenäköön sisälletään niin laitteet kuin ohjelmistot, jotka toimivat edellä mainituissa ympäristöissä. (Introduction to machine vision)

Tietokonenäkö on konenäön alalaji, jossa ulostulona on puhtaan datan sijaan kuva tai video, josta käy ilmi tulokset. Tietokonenäössä voidaan käyttää konenäkö tukemassa tai kehittäessä malleja.

2.1 Kone- ja tietokonenäön hyödyt

Kone- ja tietokonenäön suurin hyöty verrattuna ihmiseen on suuren datamäärän käsittely pienessä ajassa ja inhimillisten virheiden, kuten väsymys ja kyllästyminen välttäminen. Kone- ja tietokonenäkö voi myös automatisoida tällä hetkellä manuaalisesti hoidettuja tehtäviä laskien työtaakkaa.

2.2 Kone- ja tietokonenäön hyödyntäminen kaupan alalla

Kaupan alalla kone- ja tietokonenäöllä voidaan hyödyntää jo mahdollisesti olemassa olevien valvontakameroiden tuottamaa kuva ja video materiaalia. Tästä materiaalista voidaan tunnistaa ruuhka-aikoja, tarkastella asiakkaiden käyttäytymistä kaupan sisällä, hävikin pienentämiseen, suunnitella tehokkaampaa tuotesijoittelua ja esimerkiksi tunnistaa tarve toisen kassan avaamiseen ennen kuin ruuhka ehtii syntymään. Näillä keinoilla voidaan pienellä sijoituksella nostaa asiakastyytyväisyyttä, vähentää työntekijöiden stressiä ja luoda lisää myyntiä. (Connel, J. ym., 2013)

Valvontakameroista saatua dataa voidaan käyttää esimerkiksi asiakas tiheyden laskemiseen, jolla saadaan vastaukset seuraaviin kysymyksiin:

- Mihin asiakkaat menevät tultuaan kauppaan (ja mihin he eivät mene)?
- Miten asiakkaat käyttävät aikansa kaupassa (tarjous pöydät, konsulentit)?
- Mitkä osastot ovat suosituimpia ja mitkä alisuoriutuvat?

Näillä vastauksilla on helppoa evaluoida erilaisten tarjouksien vaikutusta, tuotesijoittelun ja konsulenttien vaikutusta asiakkaiden käyttäytymiseen ja näin ollen tehostaa kaupan tuottoa. (Ray Hartjen 2019)

2.3 Kone- ja tietokonenäön haasteet

Kone- ja tietokonenäön hyödyntämisessä kaupan alalla on monia haasteita, kuten vaihteleva ja ei-säädettävä valaistus, ihmisten liikkuminen, sekavat taustat ja pakkaantuneet ihmismassat. Olemassa olevien valvontakameroiden käyttö tuo myös mukanaan useita haasteita, kuten matala resoluutioinen kuva, oudot kuvakulmat, kontrastin huonous ja likeistä aiheuttavat häiriöt (Connel, J. ym., 2017).

Moniin näistä ongelmista ei valitettavasti voi puuttua huonontamatta asiakas kokemusta. Valvontakameroiden uusiminen parempilaatuisiksi saattaa ratkaista niistä riippuvat ongelmat, mutta saattaa käydä hyvin kalliiksi asiakkaalle.

Teknisten haasteiden lisäksi kaupan ala tuo mukanaan taloudelliset haasteet. Näitä ovat kustannusten määrä ja sijoitetun pääoman tuotto (kuinka nopeasti sijoitettu raha palautuu). Nykyajan markkinoilla sijoitusten tulee palautua yrityksille yleensä kuluvan vuoden aikana. (Connel, J. ym., 2013)

3 KÄYTETYT MENETELMÄT JA YMPÄRISTÖT

3.1 Menetelmät

Projektissa käytetyt kirjastot ja ohjelmointikielet.

3.1.1 Python - ohjelmointikieli

Python on tulkattava korkean tason ohjelmointikieli. Se soveltuu hyvin käytettäväksi skriptien luomiseen ja ketterään ohjelmistokehitykseen sen dynaamisuuden takia. Pythonin syntaksi korostaa luettavuutta ja on helppo oppia. Pythonia voi käyttää myös pelkkänä skriptikielenä.

3.1.2 OpenCV – tietokonenäkökirjasto

OpenCV on avoimen lähdekoodin tietokonenäkökirjasto, joka sisältää useita satoja eri tietokonenäköön liittyvää algoritmia. OpenCV on saatavilla C++:lle, Pythonille ja Javalle. Projektissa käytetiin OpenCV-Pythonia, joka toimii API:na OpenCV:lle.

3.1.3 Dlib – koneoppimiskirjasto

Dlib on avoimen lähdekoodin koneoppimiskirjasto, joka on toteutettu C++ kielellä. Dlib sisältää Python API:n, jolla voi ajaa sen työkaluja suoraan Python skripteillä.

3.1.4 Caffe – koneoppimiskehys

Modulaarinen koneoppimiskehys. Caffe:lla luotuja malleja voi käyttää esimerkiksi OpenCV:n kanssa objektitunnistukseen.

3.2 Ympäristöt

Projektissa käytetyt ohjelmat ja ohjelmistot

3.2.1 PyCharm – ohjelmointiympäristö

PyCharm on JetBrains:in kehittämä Pythonille suunnattu ohjelmointiympäristö. PyCharmista on saatavissa ilmainen community versio tai maksullinen enterprise versio.

3.2.2 Anaconda – tieteellinen Python-jakelu

Anaconda on ilmainen ja avoimen lähdekoodin jakelu ensisijaisesti tieteelliseen tietojenkäsittelyyn Python- ja R-ohjelmointikielillä, jonka tarkoituksena on yksinkertaistaa pakettien hallintaa ja käyttöönottoa. Pakettien ja kirjastojen versioita hallinnoi paketinhallintajärjestelmä conda.

3.2.3 Jupyter Notebook – web-applikaatio

Jupyter Notebook on avoimen lähdekoodin web-sovellus, jonka avulla voidaan luoda ja jakaa asiakirjoja, jotka sisältävät live-koodia, kuten Pythonia, mutta muitakin kieliä on tuettu, kuten R-ohjelmointikieli. Jupyter Notebook mahdollistaa esimerkiksi interaktiivisen tavan tietojenkäsittelylle sekä niiden kehitykseen ja esittelyyn, integroimalla koodin ja sen lähdöt yhteen asiakirjaan, jossa yhdistyy visualisointi, matemaattiset yhtälöt ja muut rikkaat mediat, jossa ohjelmakoodi voidaan ajaa suoraan web-käyttöliittymän kautta, ilman että käyttäjän tarvitsee asentaa omalle työkoneelleen mitään ohjelmistoa.

4 PALVELINYMPÄRISTÖN KÄYTTÖÖNOTTO

Ohjelmiston kehitystä ja ajoa varten, otettiin Tampereen Ammattikorkeakoulun pääkampuksella sijaitsevasta tieto- ja viestintätekniikan ainejärjestö SOURCE ry:n palvelinsalista käyttöön Dell Poweredge R715 palvelin, jossa oli käytössä kaksi 16-ytimistä AMD:n Opteron 6276 keskusyksikköä, 128 gigabittiä keskusmuistia ja 2.5 teratavua levytilaa sekä 10 gigabitin verkkoyhteys. Palvelimen käyttöjärjestelmäksi asennettiin Linux distribuutio Ubuntu Server 18.04 LTS (Bionic Beaver), joka valittiin seuraavin perustein:

- Laaja tuki eri kirjastoille.
- Helppokäyttöinen paketinhallintajärjestelmä.
- Suurin osa toteutuksista tehty järjestelmälle.
- Suosituin Linux distribuutio koneoppimiselle.

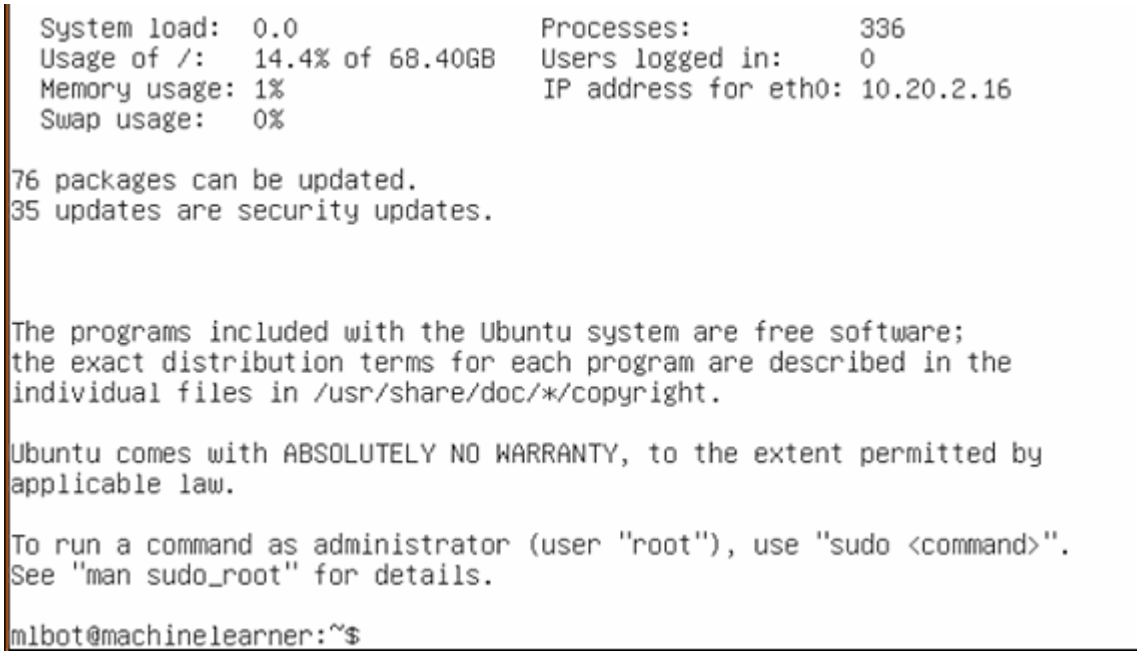
4.1 Käyttöjärjestelmän asennus

Palvelimelle asennettiin vain peruspaketit sisältävä versio Ubuntusta, joka sisältää vain välttämättömät paketit palvelimen perusasennukseen, pitäen levytilan ja keskusmuistin käytön mahdollisimman vähäisenä, vain välttämättömät peruspaketit sisältävä asennus ei sisällä graafista käyttöliittymää, vaan palvelimen hallinnointi tapahtuu komentoriviä käyttäen.

Kuvassa 1 esitellään järjestelmän asennus, jossa valittiin käyttöjärjestelmän kieleksi englanti, ja näppäimistön kieliasetukseksi Yhdysvallat. Asennettavavaksi järjestelmäversioksi valittiin Ubuntu Server, joka on yleiseen käyttöön soveltuva asennus, jota voidaan muokata tarpeen vaatiessa tilanteeseen sopivaksi palvelinjärjestelmäksi, levytilaa asetettiin järjestelmälle alustavasti 80 gigatavua.

KUVA 1. Asennuksen vaiheet

Asennuksen jälkeen kuvassa 2 näkyy järjestelmän käynnistyttyä oletuskäyttöliittymä, joka Ubuntu Server versiolla on oletusarvoisesti komentorivi, käyttäjä voi kuitenkin asentaa graafisen käyttöliittymän niin halutessaan, mutta useimmiten kyseinen toimenpide on tarpeeton palvelinympäristöissä, eikä sellaista asenneta tässä tapauksessa.



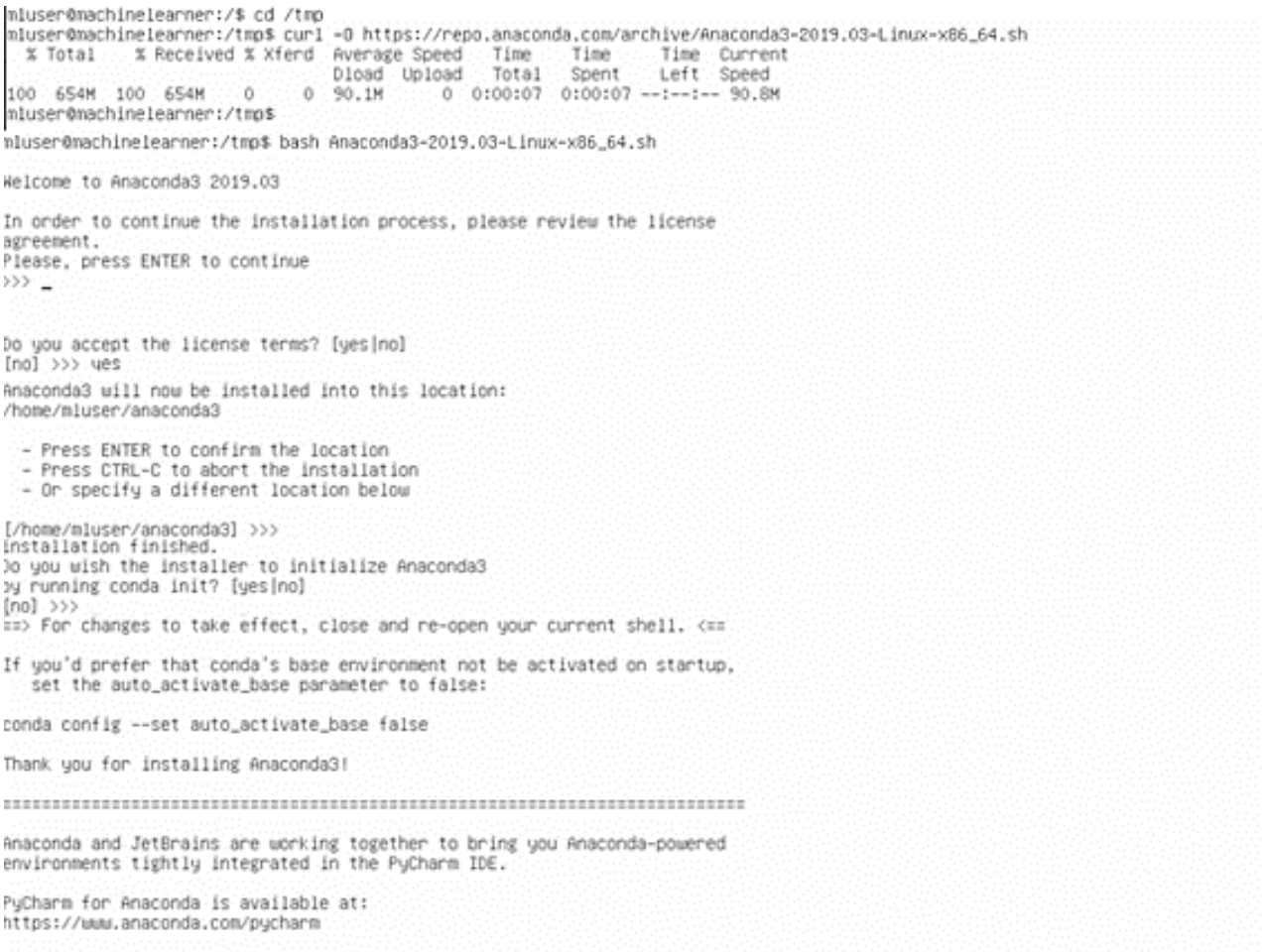
KUVA 2. Käyttöjärjestelmän oletusnäkymä

Komentorivillä näkyy järjestelmään kirjauduttua järjestelmän käyttöaste, jossa on tarkemmin esitetty prosessien määrä, keskusmuistin sekä levytilan käyttö prosentteina ja järjestelmän IP-osoite.

Järjestelmään luotiin uusi käyttäjä komennolla ”adduser <käyttäjänimi>”, jolla järjestelmän kokoonpano viimeistellään, käyttäjätilin nimeksi annettiin ”mluser” ja käyttäjälle annettiin sudo oikeudet komennolla ”usermod -aG sudo <käyttäjänimi>”. Oletusarvoisesti Linux käyttöjärjestelmissä on suositeltavaa käyttää toissijaista käyttäjätiliä, jolle on annettu sudo oikeudet järjestelmän tietoturvallisemman käytön vuoksi sekä välttäen mahdolliset konfliktit luku- ja kirjoitusoikeuksien kanssa, mikäli järjestelmällä on useampi käyttäjätili käytössä tai ajettava ohjelmisto vaatii peruskäyttäjätilin käyttöä.

4.2 Anaconda-käyttöympäristön asennus

Anaconda ohjelmistoympäristön asennus-skripti ladataan Anacondan palvelimelta curl-komennolla, jonka jälkeen ladattu asennus-skripti ajetaan bash komentotulkilla komennolla ”bash <skriptin nimi>”. Asennuksessa (kuva 3) käyttäjältä pyydetään suostumus Anacondan käyttöehtojen sekä lisenssin hyväksymiseen ja kysytään mihin asennus tulee suorittaa, oletusarvoisesti asennuspolku on käyttäjän kotihakemiston juuri, johon luodaan kansio ”Anaconda3”, mihin asentaja ajaa tarvittavat paketit. Asennuksen päätyttyä, voidaan ajaa komento ”conda init”, johon oletusarvoisesti syöte on kielteinen, mutta tässä tapauksessa ajamme komennon ”conda init” joka alustaa ohjelmiston valmiiksi käyttöä varten, tämän jälkeen tulee aloittaa uusi istunto kirjautuneella käyttäjällä asennuksen viimeistelemiseksi.



KUVA 3. Anaconda ohjelmistoympäristön asennus

Asennuksen jälkeen tarkistetaan vielä, että käyttäjälle on asennettu kaikki tarvittavat paketit ajamalla komento ”conda info” (kuva 4), josta näkee aktiivisen käyttöympäristön, mikä versio condasta ja muista riippuvista paketeista on asennettuna, sekä asennettujen pakettien repositorioiden osoitteet.

```
(base) mluser@machinelearner:~$ conda info

     active environment : base
    active env location : /home/mluser/anaconda3
            shell level : 1
       user config file : /home/mluser/.condarc
 populated config files :
          conda version : 4.6.11
    conda-build version : 3.17.8
         python version : 3.7.3.final.0
       base environment : /home/mluser/anaconda3  (writable)
           channel URLs : https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/linux-64
                          https://repo.anaconda.com/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /home/mluser/anaconda3/pkgs
                          /home/mluser/.conda/pkgs
       envs directories : /home/mluser/anaconda3/envs
                          /home/mluser/.conda/envs
               platform : linux-64
             user-agent : conda/4.6.11 requests/2.21.0 CPython/3.7.3 Linux/4.15.0-47-generic ubuntu/18.04.2 glibc/2.27
                UID:GID : 1001:1001
             netrc file : None
           offline mode : False

(base) mluser@machinelearner:~$
```

KUVA 4. Conda paketinhallintajärjestelmän tiedot

Anacondan asennuksen jälkeen, luo paketinhallintajärjestelmä (conda) ”base” virtuaaliympäristön, johon käyttäjä voi halutessaan asentaa komennolla ”conda install <paketin nimi> ” paketteja, jotka löytyvät vain ”base” virtuaaliympäristöstä, on kuitenkin suositeltavaa luoda jokaista isompaa projektia, joka hyödyntää mahdollisesti eri versioita samasta paketista, oma virtuaaliympäristö, välttäen mahdolliset ristiriidat eri versioiden ja järjestelmäpolkujen kanssa.

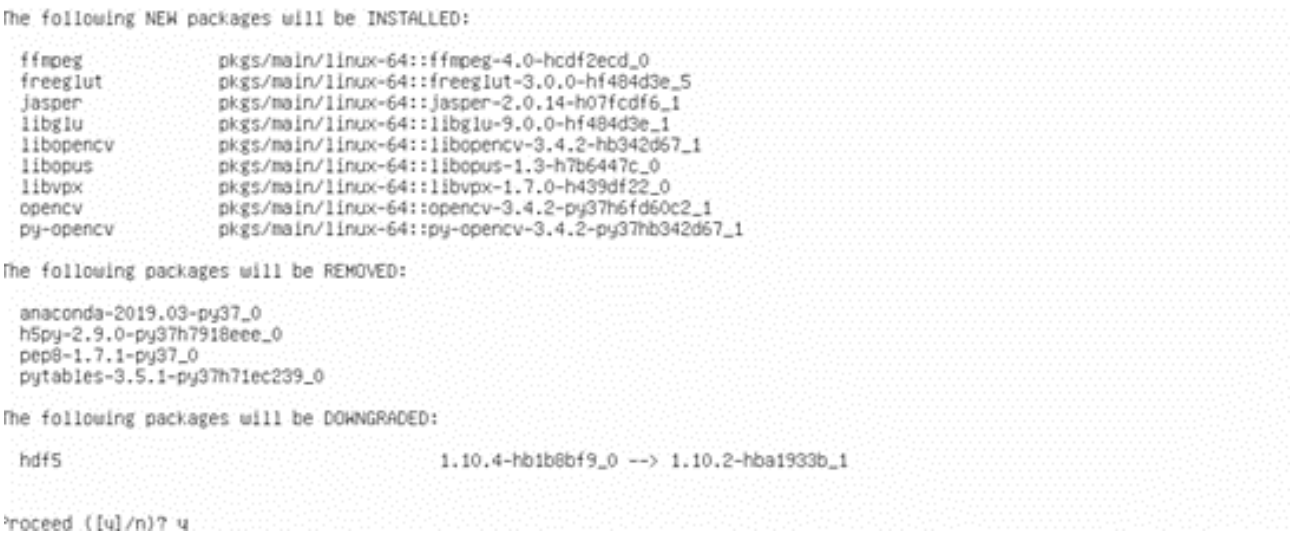
Kuvassa 5 käytetään komentoa ”conda list –explicit > spec-file.txt” listaamaan kaikki paketit, jotka kuuluvat ”base” vrituaaliympäristöön, jotka putkitetaan ” > spec-file.txt” komennon osuudella spec-file nimiseen tekstitiedostoon, joka voidaan syöttää komennolle ”conda create –name <ympäristön nimi> --file <tiedosto>” jolla luodaan tiedoston perusteella uusi virtuaaliympäristö, joka sisältää kaikki paketit, jotka on listattu spec-file tiedostossa.

```
base) mluser@machinelearner:~$ conda list --explicit > spec-file.txt
base) mluser@machinelearner:~$ conda create --name mlenv --file spec-file.txt
reparing transaction: done
erifying transaction: -
```

KUVA 5. Conda virtuaaliympäristön luonti tiedostosta

KUVA 6. Virtuaaliympäristön aktivointi ja pakettien tarkistus

Virtuaaliympäristön luonnin jälkeen, aktivoidaan uusi ympäristö komennolla ”conda activate <ympäristön nimi>” jonka jälkeen tarkistetaan kriittisimmät paketit työn kannalta ajamalla komento ”conda list <paketti>” (kuva 6). Conda esittää listana, kaikki paketit, jotka ovat joko suoraan nimellä löydettävissä, tai riippuvaisia tästä paketin nimestä ja asennettuina kyseiseen virtuaaliympäristöön, joka on asetettu aktiiviseksi. Työn kannalta tärkeimmät paketit, jotka sisältyvät Anacondan asennukseen ovat Python 3.7.3 sekä Jupyter ja siihen kuuluvat riippuvaisuudet, jotka löytyvät uudesta virtuaaliympäristöstä. Asennukseen ei kuitenkaan sisältynyt mitään ylimääräisiä kirjastoja, jonka takia suoritetaan komento ”conda install opencv”, joka asentaa OpenCV-kirjaston sekä siihen liittyvät riippuvuudet ja puuttuvat paketit (kuva 7).



KUVA 7. OpenCV-kirjaston asennus virtuaaliympäristöön

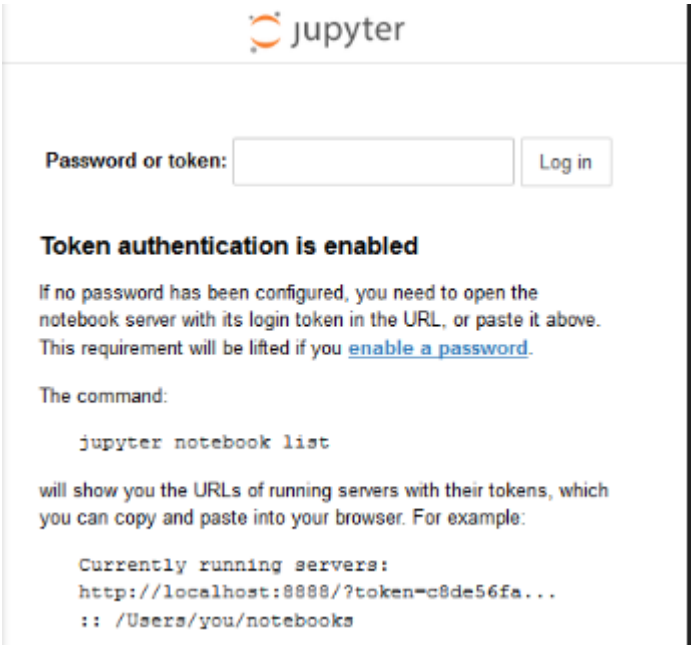
4.3 Jupyter Notebookin käyttö

Jupyter sisältyy Anacondan asennuksessa sen asentamiin peruspaketteihin oletuksena, jonka johdosta ei sen asentamisesta tarvitse huolehtia. Jupyteria voidaan käyttää suoraan web-käyttöliittymän kautta, johon navigoidaan syöttämällä selaimen osoitekenttään palvelimen IP-osoite. Ennen kuin selaimella voidaan kuitenkaan yhdistää sivulle, pitää Jupyter käynnistää ajamalla virtuaaliympäristössä komento ”conda run jupyter notebook –ip <osoite> --no-browser” joka käynnistää web-palvelimen kyseiseen osoitteeseen ja ei käynnistä palvelimen puolella selainta automaattisesti (kuva 8).

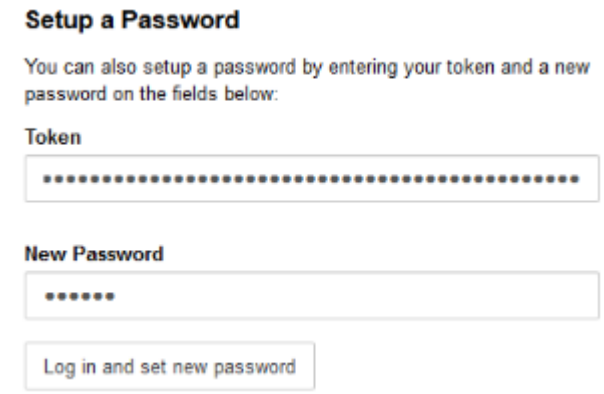
```
(base) mluser@machinelearner:~$ conda activate mlenv
(mlenv) mluser@machinelearner:~$ conda run jupyter notebook --ip 10.20.2.16 --no-browser
```

KUVA 8. Jupyter notebookin käynnistys

Koska tunnistautumista ei ole asetettu, järjestelmä oletusarvoisesti ohjaa sivulle jossa (kuvat 9 ja 10) tunnistaudumme, joko syöttämällä konsoliin tulostetun avaimen, tai käytämme avainta luodaksemme salasanan, jolla voi jatkossa tunnistautua järjestelmään.



KUVA 9. Jupyter notebookiin tunnistautuminen avaimella

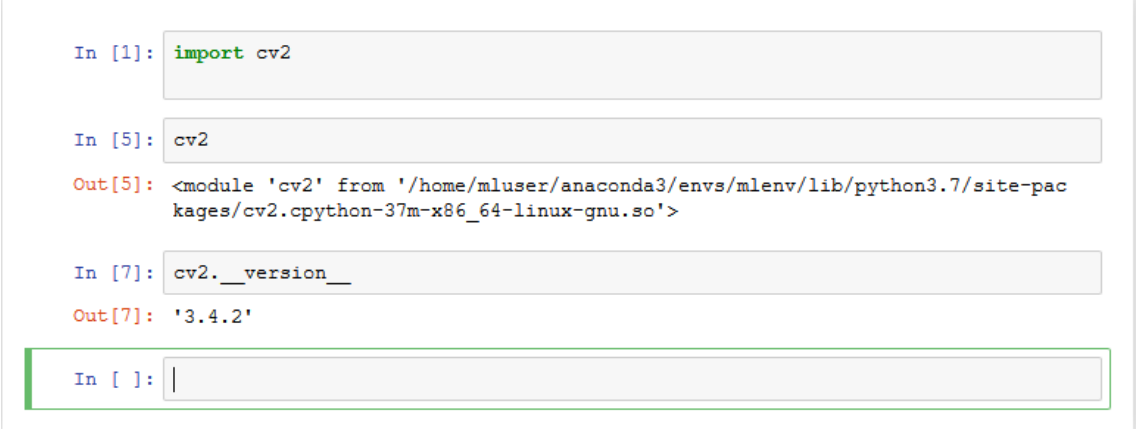


KUVA 10. Jupyter notebookiin salasanan luonti avaimella

Tunnistautumisen jälkeen, avautuu käyttäjälle perusnäkymä, jossa käyttäjä voi selata kansioita, luoda uusia kansioita tai Python-skriptejä (kuva 11).

KUVA 11. Jupyter notebookin perusnäkymä

Perusnäkymään saavuttuamme, luomme uuden kansion, johon luomme uuden Python-skripti notebookin, josta ajamme vielä OpenCV-kirjastoa (kuva 12) sen verran, että tarkistamme version, varmistuaksemme siitä, että asennus on tapahtunut oikein eikä konfigurointeja tarvitse jatkaa sen enempää.



KUVA 12. OpenCV-kirjaston ajo Jupyterissa

4.4 Dlib-kirjaston asennus ja kääntäminen

Esikäännettyä Pythonille tehtyjä binäärejä ei Dlib-kirjastosta löydy condan repositoriosta, joten se tulee kääntää koneella ja asentaa sille tarvittava Python wrapperi.

Varmistetaan ensin, että tarvittavat käyttöjärjestelmän kirjastot ovat asennettuna ja asennetaan ne, mikäli niitä ei ole (kuva 13).

```
(mlenv) mluser@machinelearner:~$ sudo apt-get install build-essential cmake
pkg-config libx11-dev libatlas-base-dev libgtk-3-dev libboost-python-dev
```

KUVA 13. Tarvittavat käyttöjärjestelmän kirjastot

Kun käyttöjärjestelmään on asennettu tarvittavat kirjastot kääntämistä varten, voimme ladata Dlib-kirjaston Dlib.net osoitteesta, asennettava versio riippuu mitä muut kirjastot tukevat, oletuksena otimme sen hetkisen uusimman stabiilin version kirjastosta, joka oli 19.6 -versio.

Kirjasto käännetään ja linkataan kuvien 14 ja 15 mukaisesti, jossa kuvassa 14 ajamme ”cmake ..” komennon, jossa määritämme lähdekansion. Kuvassa 15 ”cmake” komennolle annetaan parametrit ”--build <polku> --config <konfiguraatio>” ja komennolla ”make install” asennamme kyseisen binäärin, jonka jälkeen suoritetaan ”ldconfig” komento, joka luo tarvittavat linkit ja säilyttää välimuistissa kaikkein uusimmat jaetut kirjastot siitä kansiosta mikä on määritetty kehotteessa tai nykyisen kansion, jossa komento suoritetaan.

```
(mlenv) mluser@machinelearner:~$ cd dlib-19.6/
(mlenv) mluser@machinelearner:~/dlib-19.6$ mkdir build
(mlenv) mluser@machinelearner:~/dlib-19.6$ cd build
(mlenv) mluser@machinelearner:~/dlib-19.6/build$ cmake ..
```

KUVA 14. cmake lähde

```
(mlenv) mluser@machinelearner:~/dlib-19.6/build$ cmake --build . --config Release
Scanning dependencies of target dlib
[  0%] Building CXX object dlib/CMakeFiles/dlib.dir/base64/base64_kernel_1.cpp.o
[  1%] Building CXX object dlib/CMakeFiles/dlib.dir/bigint/bigint_kernel_1.cpp.o
[  1%] Building CXX object dlib/CMakeFiles/dlib.dir/bigint/bigint_kernel_2.cpp.o
[  2%] Building CXX object dlib/CMakeFiles/dlib.dir/bit_stream/bit_stream_kernel_1.cpp.o
[  2%] Building CXX object dlib/CMakeFiles/dlib.dir/entropy_decoder/entropy_decoder_kernel_1.cpp.o
[  3%] Building CXX object dlib/CMakeFiles/dlib.dir/entropy_decoder/entropy_decoder_kernel_2.cpp.o
[  3%] Building CXX object dlib/CMakeFiles/dlib.dir/entropy_encoder/entropy_encoder_kernel_1.cpp.o
[  4%] Building CXX object dlib/CMakeFiles/dlib.dir/entropy_encoder/entropy_encoder_kernel_2.cpp.o
[  4%] Building CXX object dlib/CMakeFiles/dlib.dir/md5/md5_kernel_1.cpp.o
[  5%] Building CXX object dlib/CMakeFiles/dlib.dir/tokenizer/tokenizer_kernel_1.cpp.o
[  5%] Building CXX object dlib/CMakeFiles/dlib.dir/unicode/unicode.cpp.o
[  6%] Building CXX object dlib/CMakeFiles/dlib.dir/data_io/image_dataset_metadata.cpp.o
```

KUVA 15. tiedostojen koonti ja kääntö

Kun edellinen vaihe on suoritettu, voimme ajaa Dlib-19.6 kansion juuressa setup.py tiedoston komennolla ”python <tiedosto> install” (kuva 16), joka luo Pythonille wrapperin, jonka kautta voidaan Dlib- kirjastoa käyttää Jupyterin kautta.

```
reating /home/mluser/anaconda3/envs/mlenv/lib/python3.7/site-packages/dlib-19.6.0-py3.7-linux-x86_64.egg
xtracting dlib-19.6.0-py3.7-linux-x86_64.egg to /home/mluser/anaconda3/envs/mlenv/lib/python3.7/site-packages
dding dlib 19.6.0 to easy-install.pth file

nstalled /home/mluser/anaconda3/envs/mlenv/lib/python3.7/site-packages/dlib-19.6.0-py3.7-linux-x86_64.egg
rocessing dependencies for dlib==19.6.0
inished processing dependencies for dlib==19.6.0
(mlenv) mluser@machinelearner:~/dlib-19.6$
```

KUVA 16. setup.py -tiedoston ajo

Ajon aikana ei tullut ongelmia, joten voimme kokeilla tuoda Dlib-kirjaston ja tarkistaa sen version ja todeta että asennus onnistui (kuva 17).

```
In [4]: import dlib
        dlib.__version__

Out[4]: '19.6.0'
```

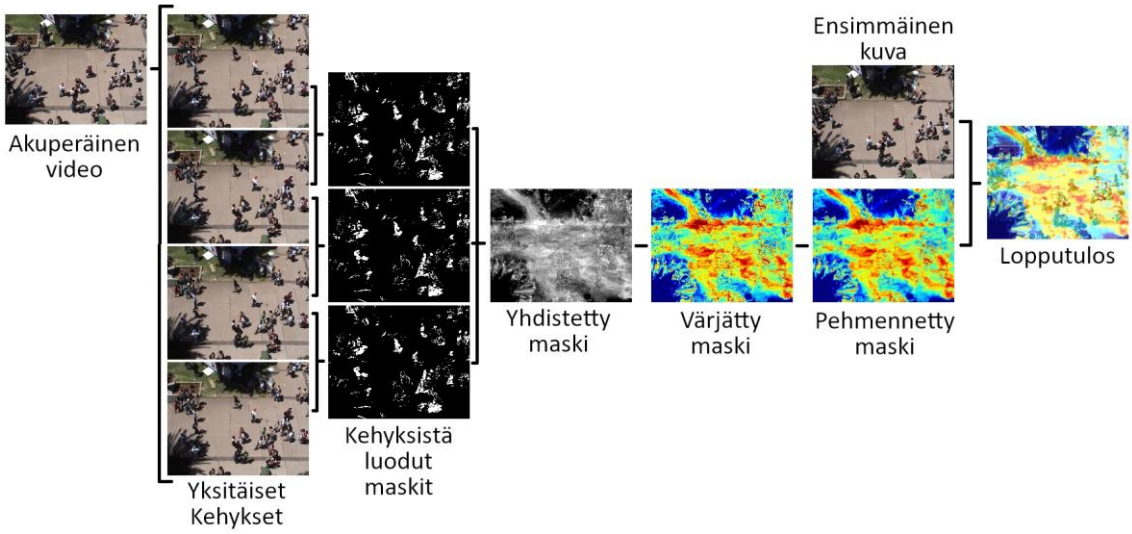
KUVA 17. Dlib-kirjaston version tarkistus

5 OPENCV JA TIHEYSLASKENTA

Ihmismassojen tiheyskartoitus OpenCV:n Python-kirjaston avulla. Ohjelmalle annetiin video, josta se loi ihmisten liikkeistä tiheyskartan. Tiheyskartassa Ihmisten liikkuminen näkyy valituilla värisävyillä. Ohjelma koodi löytyy liitteestä 1.

Ohjelman pohjana käytettiin Intelin julkaisemaa OpenCV esimerkki ohjelmaa (Intel, 2018). Ohjelma suoritettiin palvelimella käyttäen Jupyter Notebookkia.

5.1 Ohjelman toiminta



KUVA 17. Ohjelman toiminta

Ohjelma lataa videon, josta se erottelee jokaisen yksittäisen kehyksen ja suorittaa niille liikkeen tunnistuksen. Liikkeen tunnistamiseen ohjelma käyttää OpenCV:n treshold metodin binary_treshold moodia. Treshold metodi tunnistaa liikkeen edellisestä kehyksestä kynnysarvon avulla ja nostaa sen haluttuun arvoon luoden maskin jokaisesta kehys parista. Metodilta tulleet kehyksen lisätään toisiinsa, joka luo yhdistetyn maskin, josta näkyy liikkeen määrä videossa valkoisella. Mustavalkoinen maski värjätään OpenCV:n applyColorMap funktiolla haluttuun värikarttaan. Värjätty maski käytetään vielä OpenCV:n blur funktion läpi, joka pehmentää kuvan käyttäen keskiarvoa annetulla kerneli-koolla. Lopputulos saadaan lisäämällä pehmennetty maski OpenCV:n addWeighted

funktiolla videon ensimmäiseen kuvaan paino arvolla 0.7. Kuvassa 17 ohjelman toiminta visuaalisesti esitettynä.

5.2 Ohjelman parametrit

Ohjelmalle voi antaa parametreinä video tiedoston, frame skipin, liikkeen kynnysrajan, kehyksen arvon, käytettävän värikartan ja kehyksen sumennuksen (kuva 18).

```
"""
Parameters for heatmap generation
cap = Video source (if using live feed main loop must be tweaked)
step = Frameskip
tresh = Binary treshold for setting the result to maxValue
maxValue = amount of motion to be picked up 1 = least and bigger value means more motion captured
colormap = OpenCV colormap to use JET is simple and goodlooking HOT is also usable but less clear
blur = Ammount of blur in heatmap. Makes it easier to look at
"""
cap = cv2.VideoCapture('students003.avi')
step = 1
thresh = 2
maxValue = 2
colormap = cv2.COLORMAP_JET
blur = 1
```

KUVA 18. Tiheyskartan parametrit

5.2.1 Video tiedosto (cap)

Video tiedostolla valitaan video lähde, josta tiheyskartta luodaan. Tiedoston sijaan voi myös ohjelmalle ohjata live kuvaa joltain koneessa kiinni olevasta lähteestä. Live kuvan käsittelyä varten tulee ohjelmassa tehdä muutoksia pää silmukaan.

5.2.2 Frame skip (step)

Frame skipillä voidaan ohjelma laittaa suorittamaan vain joka n:nes kehys, joka keventää huomattavasti ohjelman ajoa ja pitkissä videoissa parantaa luettavuutta vähentämällä peräkkäisten kehysten päällekkäisyyttä.

5.2.3 Liikkeen kynnysraja (tresh)

Liikkeen kynnysraja asettaa ohjelmalle kynnysarvon, jolla se tunnistaa muutosta kuvasta. Jos kahden kehyksen välillä pikselin valoisuuden muutos ylittää annetun kynnysrajan se rekisteröidään liikkeeksi ja nostetaan kehyksen maksimiarvoksi.

5.2.4 Kehyksen arvo (maxValue)

Kehyksen arvo määrittää yhden kehyksen painoarvon ulostulo kuvassa. Jokaisesta kehyksestä tunnistetaan liike liikkeen kynnysrajaa käyttäen ja niiden painoarvo kerätään yhteen kehykseen asteikolla 0 – 255. Kehyksen arvoa säätämällä voidaan vaikuttaa kuinka paljon liike ”värjää” ulostulo kuvaa.

5.2.5 Värikartta (colormap)

Värikartalla asetetaan haluttu värimaailma ulostulo kuvalle. OpenCV:stä löytyy useita värikarttoja, mutta vain muutamat niistä soveltuvat käytettäväksi tiheyskartassa. Näistä parhaiksi todettiin COLORMAP_JET ja COLORMAP_HOT.

5.2.6 Kehyksen sumennus (blur)

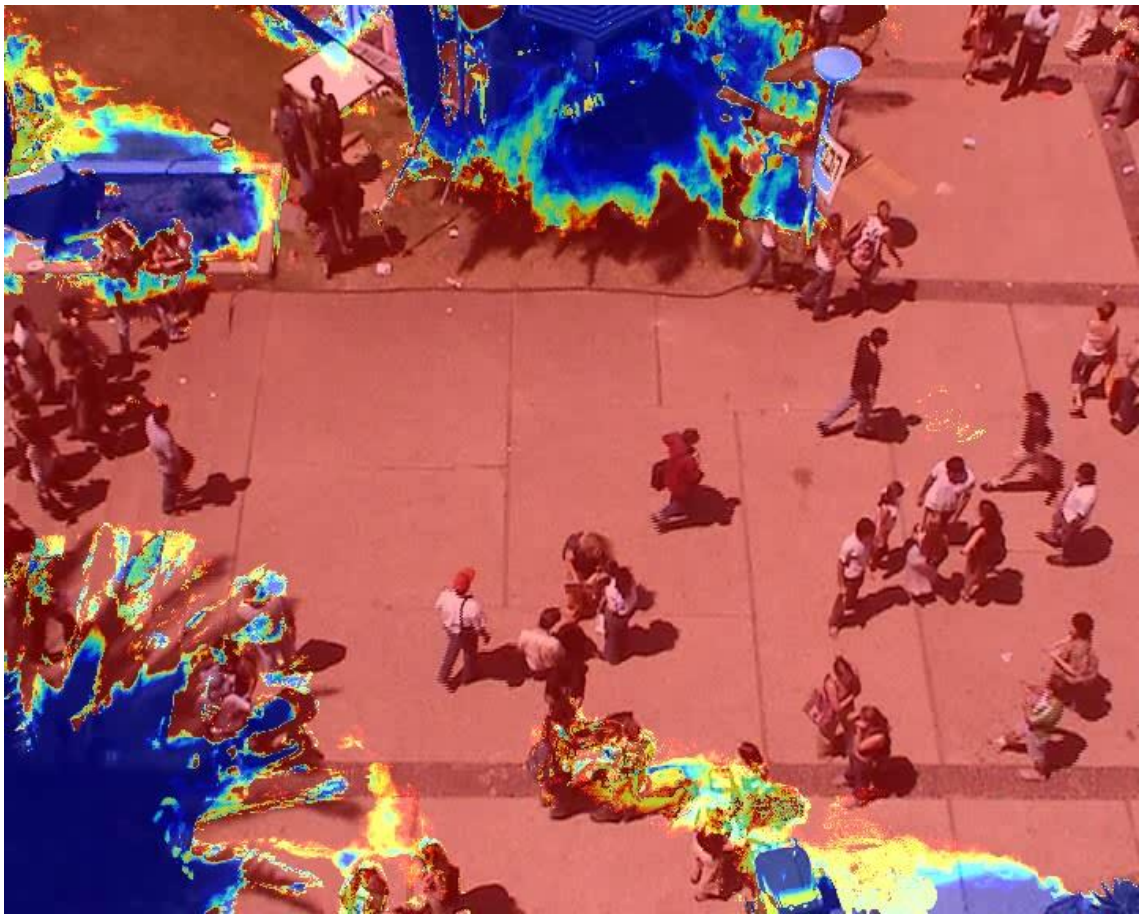
Kehyksen sumennus määrittää kerneli-koon, jolla jokaiselle pikselille lasketaan keskiarvo ympärillä olevista pikseleistä. Tämä tekee kuvasta huomattavasti mukavamman katsella.

5.3 Ohjelman ajo ja optimointi

Ohjelmalle annetavia parametrejä tulee muokata lähde videon mukaan, jotta lopputulos on hyvännäköinen. Videon ollessa hyvin pitkä tulee frame skippiä nostaa ja kehyksen arvoa pienentää, jotta kuva ei ylipainota liikettä. Jos taas video on lyhyt ei frame skippiä tarvita ja kehyksen arvoa saattaa joutua nostamaan, jotta vähäinen liikekin tunnistetaan.

Ajoimme ohjelmaa valitulla testivideollamme eri parametreillä asettaen aluksi parametrit samoiksi kuin kuvassa 18. Seuraavia ajoja varten muokkasimme parametrejä parantaaksemme kuvan luettavuutta.

5.3.1 Ensimmäinen ajo



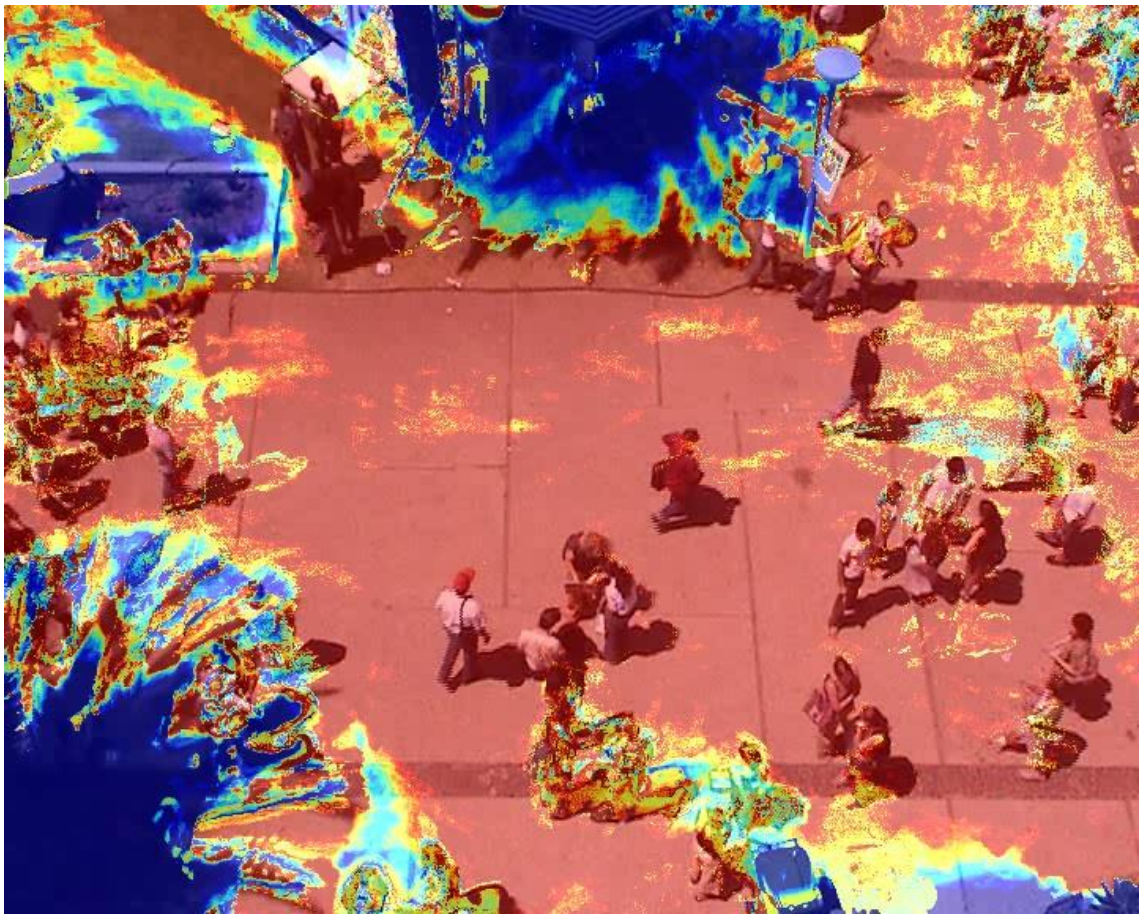
KUVA 19. Ensimmäisen ajon ulostulo

Ensimmäinen ajo ajettiin taulukossa 1 olevilla parametreillä. Tuloksista (kuva 19) huomatiin, että kuva ylipainottaa liikkeen määrää luoden alueesta missä liikettä esiintyy kokonaan punaisen jättäen vain muutaman kohdan muun väriseksi.

TAULUKKO 1. Ensimmäisen ajon parametrit

Parametri	**Arvo**
step	1
threshold	2
maxValue	2
colormap	COLORMAP_JET
blur	1

5.3.2 Toinen ajo



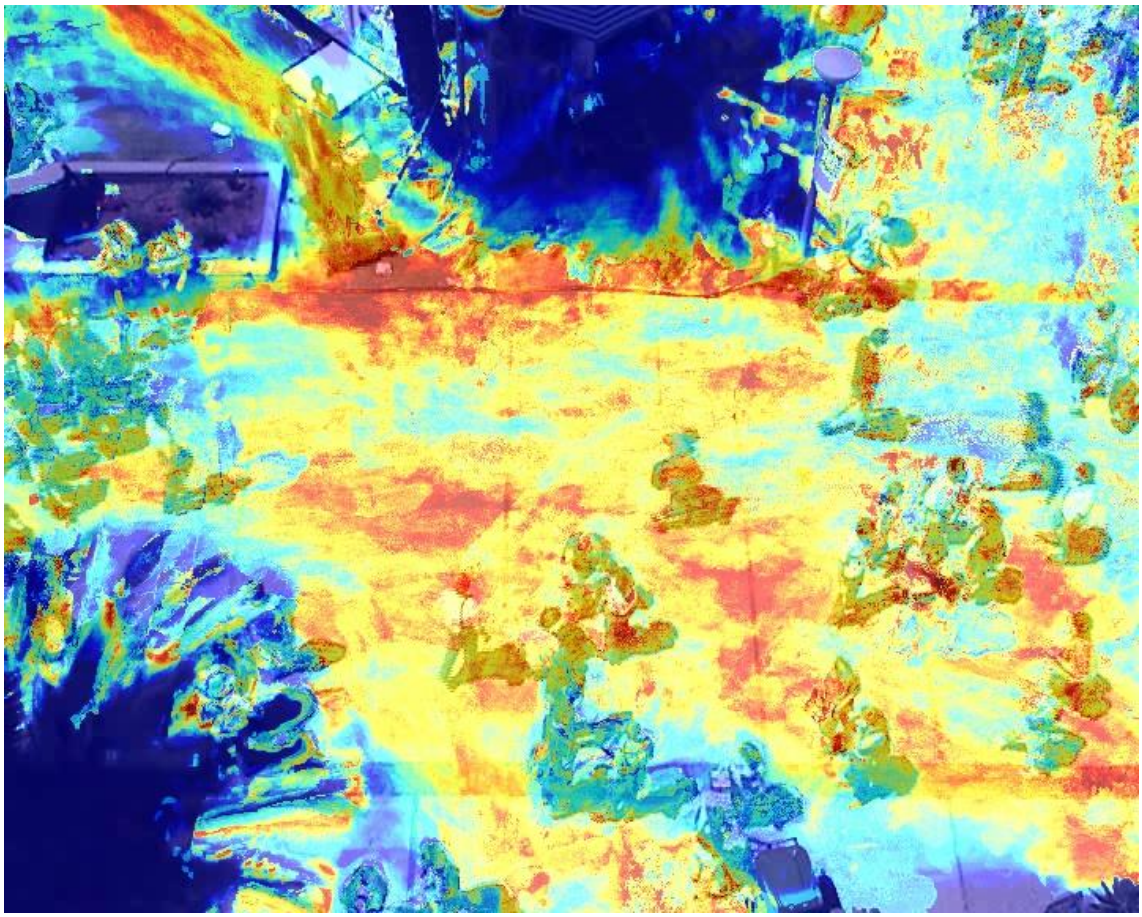
KUVA 20. Toisen ajon ulostulo

Toiseen ajoon lisätiin frame skippiä, joka ajon aikana jättää huomioimatta puolet videon kehyksistä (taulukko 2). Tällä saatiin vähän parannettu liikkeen päällekkäisyyttä (kuva 20). Lopputulos oli jo vähän parempi, mutta siinä oli siltikin vielä parantamisen varaa.

TAULUKKO 2. Toisen ajon parametrit

Parametri	**Arvo**
step	2
threshold	2
maxValue	2
colormap	COLORMAP_JET
blur	1

5.3.3 Kolmas ajo



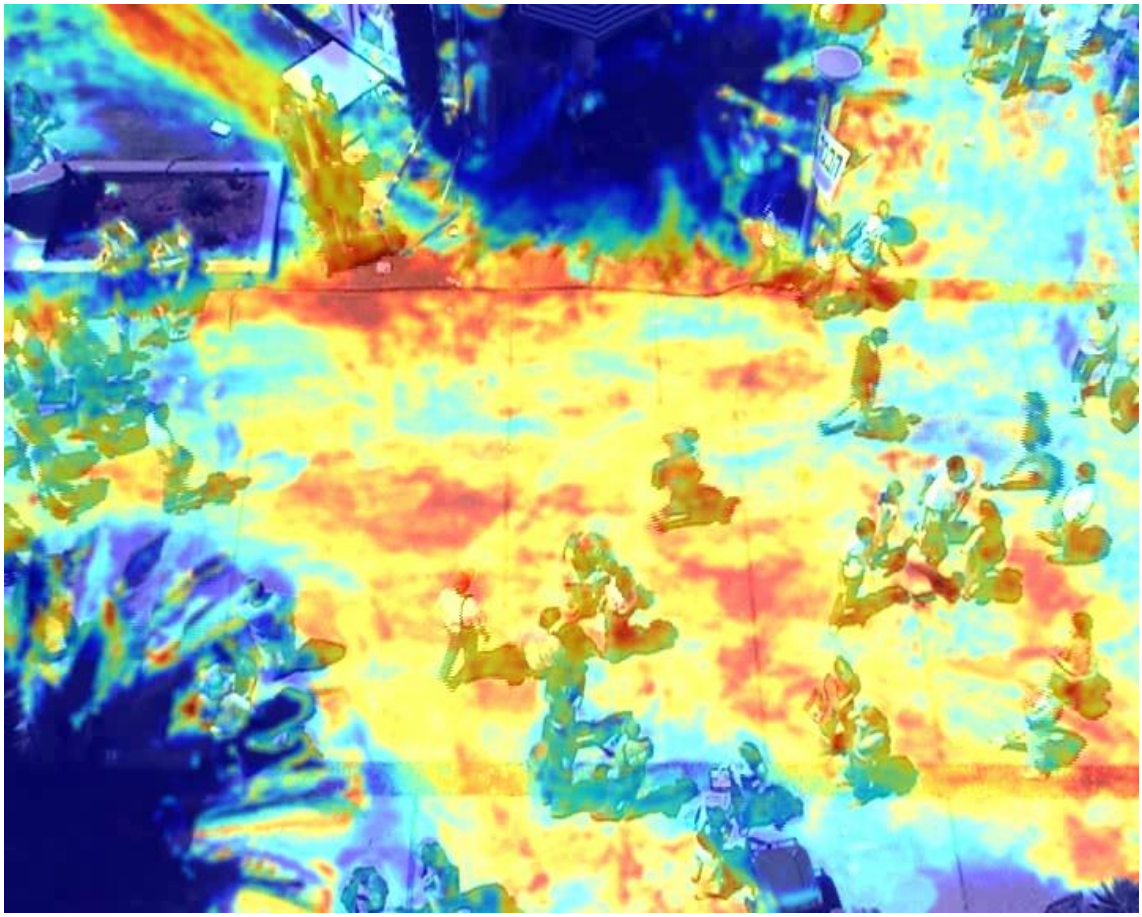
KUVA 21. Kolmannen ajon ulostulo

Kolmanteen ajoon asetettiin kehyksen arvoksi 1 (taulukko 3), joka huomattavasti vähensi jokaisen kehyksen painoa lopputuloksessa. Tuloksesta (kuva 21) voitiin todeta, että liikkeen tunnistus alkoi olemaan jo halutulla tasolla, mutta kuva ei ollut vielä hyvin luettava värien rosoisuuden takia.

TAULUKKO 3. Kolmannen ajon parametrit

Parametri	**Arvo**
step	2
threshold	2
maxValue	1
colormap	COLORMAP_JET
blur	1

5.3.4 Neljäs ajo



KUVA 22. Neljännen ajon ulostulo

Neljänteen ajoon lisättiin sumennus (taulukko 4), joka loi tuloksesta (kuva 22) huomattavasti mukavamman lukea. Tuloksesta erottaa selvästi missä kohtaa kuvaa on ollut liikettä ja missä sitä on ollut eniten.

TAULUKKO 4. Neljännen ajon parametrit

Parametri	**Arvo**
step	2
threshold	2
maxValue	1
colormap	COLORMAP_JET
blur	5

5.3.5 Viides ajo

KUVA 23. Neljännen ajon ulostulo

Viidennessä ajossa kokeiltiin värikarttaa COLORMAP_HOT (taulukko 5). Tuloksesta (kuva 23) näkee edelleen liikkeen määrän selvästi ja värikartan valinta onkin lähinnä kiinni käyttäjän mieltymyksestä. COLORMAP_HOT sopii yleisesti paremmin tumman sävyisiin videoihin, kun taas COLORMAP_JET soveltuu vaaleampiin videoihin.

TAULUKKO 5. Viidennen ajon parametrit

Parametri	**Arvo**
step	2
threshold	2
maxValue	1
colormap	COLORMAP_HOT
blur	5

5.4 Ajojen purkaminen

Ajojen tuloksista huomatiin eri parametrit vaikutus lopputulokseen ja viimeisen kahden ajon tulokset olivatkin jo täysin käytettäviä kartoituksia ihmisten liikkeestä videon kuvaamalla alueella.

5.4.1 Ongelmat

Suurin ongelma ohjelmassa on parametrien erilainen vaikutus eri videoihin. Pelkkä videon pituuden muutos aiheuttaa tulosten muuttumisen vertailukelvottomiksi. Tämä tekee analysoinnista kyseisellä ohjelmalla hieman työlästä, mutta toisaalta asetukset tarvitsevat säätää vain kerran oikeiksi.

Toistuvien ympäristöjen liikkeiden tunnistus saattaa vääristää tuloksia joissain tapauksissa, kuten esimerkkivideolla kamera tunnistaa palmujen heilunnan tuulessa liikkeeksi. Tämä ei välttämättä ole niin iso ongelma sisätiloissa, mutta se tulee silti muistaa tuloksia tarkastellessa.

5.4.2 Käyttökohteet

Tiheyskartoilla pystyy tunnistamaan ruuhkakohtia kaupassa ja näin ollen keskittämään mainostusta ja tuotteita myyvämmille paikoille. Tiheyskarttojen avulla voi myös seurata konsulenttien ja tarjousten vaikutusta ihmisten liikkumiseen kaupassa.

6 OPENCV, DLIB, CAFFE JA IHMISLASKURI

Ohjelma tunnistaa videolta tai kamerakuvasta ihmiset ja seuraa niiden liikettä. Se myös laskee kuvan keskikohdan ylittäneiden ihmisten lukumäärän ja laskee, kuinka monta ihmistä on liikkunut vasemmalle tai oikealle.

Ohjelman ulostulo on video tiedosto, jossa näkyy vihreillä pisteillä tunnistetut ihmiset, ihmisten tunnisteet, keskikohta ja laskuri.

Ohjelma on alun perin Adrian Rosebrock:in julkaisema blogissaan PyImageSearch. Ohjelmaan tehtiin muutoksia, jotta se sopisi videomateriaalimme käsittelyyn ja sitä suoritettiin palvelimella käyttäen Jupyter Notebookkia.

6.1 Ohjelman toiminta

Ohjelman toiminta voidaan jakaa kolmeen päätehtävään: ihmisten tunnistus, seuraus ja laskeminen. Ohjelma näyttää myös ajon aikana videon tai kamerakuvan, jota sen läpi ajetaan.

6.1.1 Ihmisten tunnistus

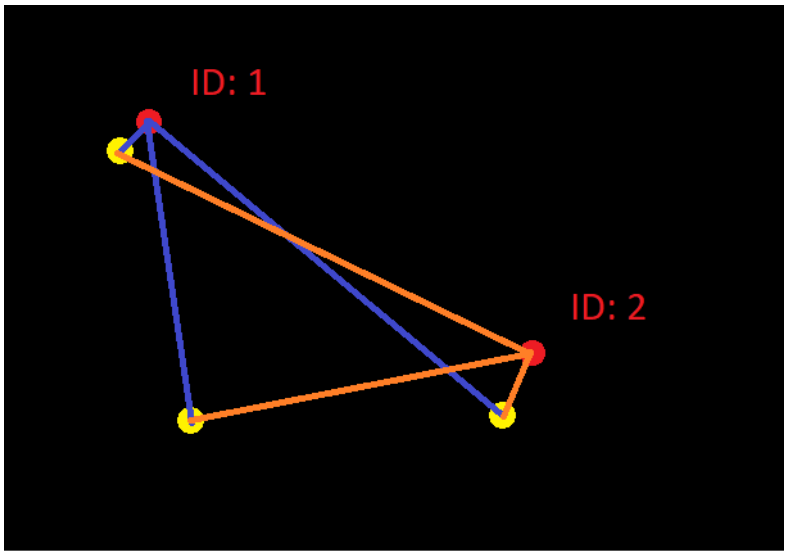
Ohjelma käyttää ihmisten tunnistamiseen kuvasta OpenCV:n Deep Neural Network moduulia. Kyseisellä moduulilla luodaan neuroverkko Caffe mallin avulla, joka sisältää hermo verkko mallin ihmisten tunnistusta varten.

Videon jokainen kehys ajetaan Deep Neural Network moduulin blobFromImage metodin läpi, joka luo siitä BLOB:in (Eng. binary large object). BLOB on binäärimuotoinen objekti, joka sisältää kuvasta löytyneet ”kiinnostavat” muodot.

Kehyksestä muodostettu BLOB asetetaan sisääntuloksi neuroverkolle, joka lajittelee siitä löytyvät muodot sen mukaan mitä se arvelee niiden olevan ja palauttaa kohteen keskipisteen ja varmuuden millä esine tai asia luokiteltiin. Tunnistetut kohteet lisätään listaan, josta ne annetaan seuraimelle.

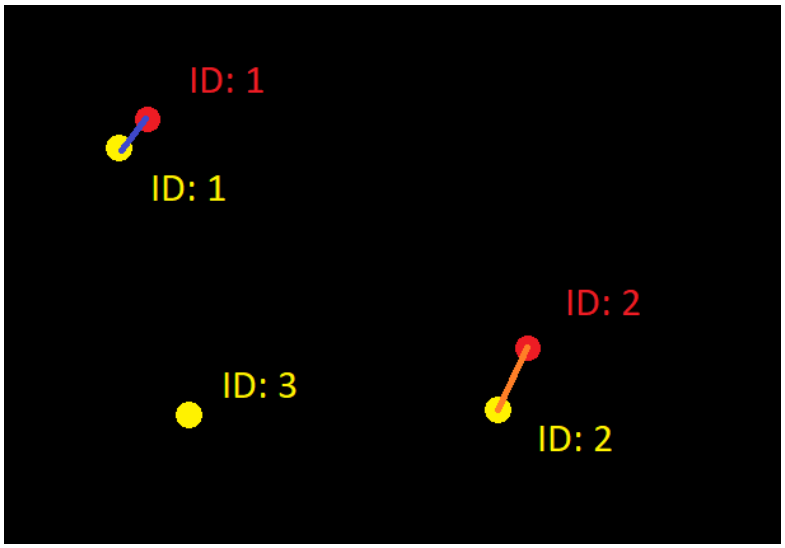
6.1.2 Ihmisten seuraus

Ihmisten seuraus saa tunnistukselta listan ihmisistä, joita on havaittu kehyksessä. Kuvassa 24 on kuvattu kuinka seuranta mittaa edellisissä kehyksissä havaittujen pisteiden etäisyyttä uudessa kehyksessä lisättyihin pisteisiin. Ohjelma käyttää keskipisteiden seurantaan euklidista etäisyyden mittausta.



KUVA 24: Pisteiden välisten matkojen arvio

Jos kahden pisteen matka on tarpeeksi lyhyt, annetaan sille sama ID kuin lähimmäisellä pisteellä. Jos pisteelle ei löydy alkuperää viimeisestä neljästäkymmenestä kehyksestä annetaan sille uusi ID (kuva 25).

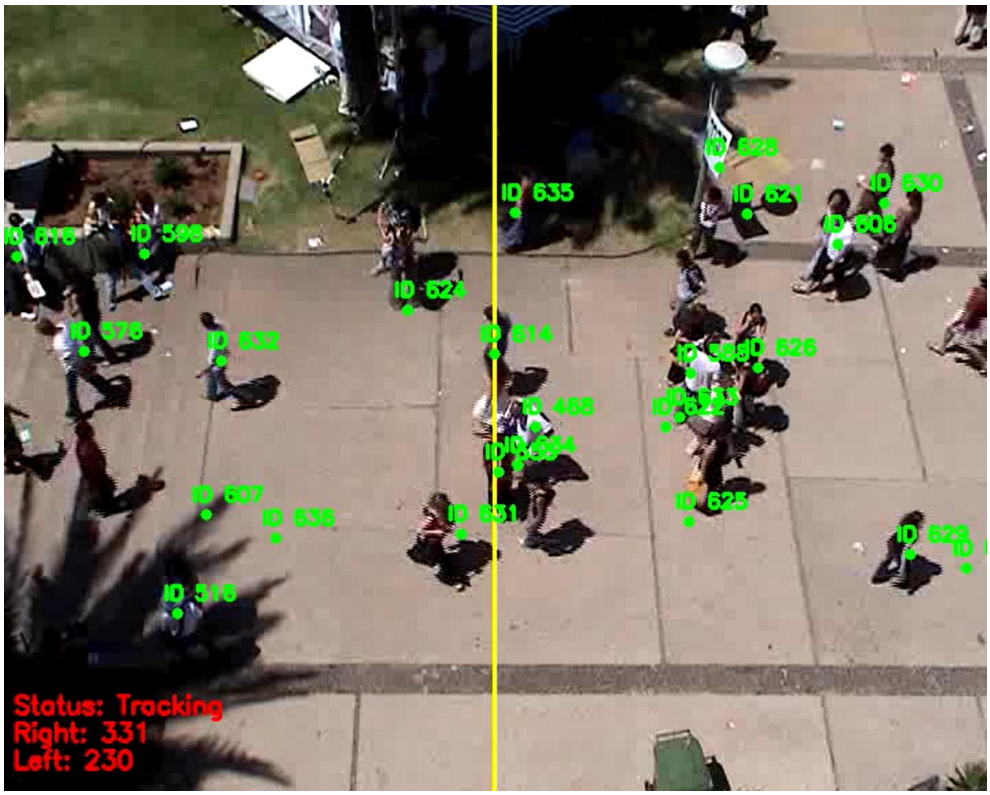


KUVA 25: ID:n määritys

6.1.3 Ihmisten laskeminen

Ihmisten seurauksen aikana niistä tallennetaan myös keskipisteiden x-akselilla tapahtuvan muutoksen suunta. Keskipisteiden ylittäessä kehyksen keskikohta katsotaan, onko samalla ID:llä oleva piste ohittanut vielä keskikohtaa ja mikä sen liikkeen suunta on. Jos kyseisellä ID:llä oleva piste ei ole vielä ylittänyt keskikohtaa lisätään se sijainnin ja liikkeen suunnan perusteella joko vasemmalle tai oikealle liikkuviin.

Kuvassa 26 käyttöliittymä, johon piirretty keltaisella viivalla kohta, jonka ylittäessä lasketaan henkilön ylittäneen keskikohdan, sekä vasemmassa alakulmassa laskuri, joka laskee kumpaankin suuntaan menevät ihmiset.



KUVA 26: Ihmisten laskenta

6.2 Ohjelmaan tehdyt muutokset

Ohjelmaan joutui videon ajoa varten tekemään muutaman muutoksen. blobFromImage metodin tarkkuutta tuli nostaa pienentämällä RGB kanavien normalisointia. RGB kanavien normalisoinnin poisto hidastaa huomattavasti ajamista, mutta koska meillä on käytössä geneerinen konenäkö malli se parantaa ihmisten tunnistettavuutta.

Toinen muutettu asia oli laskuri. Laskuri vaihdettiin seuraamaan liikettä x- akselia y- akselin sijaan. Myös käyttöliittymä vaihdetiin näyttämään keskiviiva horisontaalisesti ja laskurien tekstit heijastamaan muutoksia.

6.3 Ohjelman ajaminen

Ohjelmaa ajettiin seuraavilla argumenteilla:

--confidece 10
--prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt
--model mobilenet_ssd/MobileNetSSD_deploy.caffemodel
--input student003.avi
--output output/output_01.avi

Confidence argumentti pienentää pienintä hyväksyttävää varmuutta, jolla tunnistus suoritetaan.

Protxt on polku Caffe:n käyttöönotto tiedostoon, joka sisältää Caffe mallin tarvitsemat asetukset

Model on polku käytettävään Caffe mallliin

Input on sisääntulo, joka voi olla video tiedosto tai live kamerakuva.

Output on ulostulo tiedosto, joka on käytännössä sama video, mikä näkyi ajaessa.

6.4 Tulokset

6.4.1 Ongelmat

Ohjelma sai videon huonohkosta laadusta huolimatta seurattua kohtalaisen hyvin ihmisten liikettä ruuhkaisella kadulla. Eniten ongelmia oli ryhmänä kulkevien ihmisten tunnistamisessa ja ihmisten liikkuessa erilaisten taustojen välillä.

Laskimelle ongelmallista oli kameran katselukulman suuruus, jonka takia videolla ihmiset saattoivat liikkua useampaan suuntaan kuin pelkästään vasemmalle tai oikealle videon aikana. Tämä saattaa hämätä laskuria laskemaan ihmisten kävelysuunnan väärin. Tulosten luotettavuutta parantaisi kuvattavan alueen rajaaminen pienemmäksi, tasaisempi valaistus ja homogeeninen tausta.

6.4.2 Käyttökohteet

Kaupan alalla henkilö laskurilla pystyy helposti laskemaan sisään ja ulos menevien asiakkaiden määrän, sekä havaitsemaan ruuhkatilanteita. Tunnistus alueen määrittäminen on helppo muokata esimerkiksi neliöksi ja laskea aikoja, mitä ihmiset viettävät hyllyjen, konsulenttien ja tiettyjen tuotteiden kohdalla.

7 POHDINTA JA JATKOKEHITYS

Kaupan alalla on monia kohteita, jossa konenäköä voisi hyödyntää markkinoinnin ja muun toiminnan tukena. Ihmisten kaupassa liikkumisen analysointi on halpa ja helppo toteuttaa käyttäen jo olemassa olevia kamera järjestelyitä. Analysointia voisi yrityksille tarjota joko tuotteena tai palveluna riippuen asiakaan tarpeista ja yrityksen resursseista.

Kumpikin työssä läpi käyty menetelmä soveltuu suoraan markkinoinnin parantamiseen kaupan alalla. Tiheyskartan avulla voi kauppias tai myymälävastaava suunnitella mainostuksen ja tuotesijoittelun vastaamaan moderneja tarpeita. Ihmislaskurilla voi taas tarkastella ihmisten kulkua eri alueilla, esimerkiksi laskemalla käytävillä kulkevat ihmiset ja tarkastella kuinka kauan ihmiset viettävät tietyissä hyllykohdissa. Ihmislaskuri mahdollistaa myös reaaliaikaisen tarkastelun, jolla pystytään tunnistamaan syntyviä ruuhkia ja evaluoida kassojen tarvetta.

7.1 Jatkokehitys

Ohjelmien ajamiseen tulisi kehittää käyttöliittymä, jolla niiden parametrien antaminen onnistuisi helpommin ja ohjelman käyttäjän ei tarvitsisi muuntaa ohjelmakoodia käyttäessään niitä. Käyttöliittymän voisi toteuttaa mahdollisesti web-käyttöliittymänä tai Python sovelluksena. Käyttöliittymään voisi myös lisätä ominaisuudet ihmistunnistus alueiden määrittämiseen graafisesti.

LÄHTEET

Alon Learner. University Students. Ladattu 1.4.2019. https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data

Cognex. Introduction to machine vision. Luettu 7.5.2019. https://www.assembly-mag.com/ext/resources/White_Papers/Sep16/Introduction-to-Machine-Vision.pdf

Ray Hartjen. 2018. Computer Vision Sees Better Than 20/20. Julkaistu 8.4.2019 Luettu 7.5.2019 https://retailnext.net/en/blog/computer-vision-sees-better-than-2020/

Connel, J., Fan, Q., Gabbur, P., Haas, N., Pankanti, S & Trinh, H. Retail Video Analytics: An Overview and Survey. 2013 https://www.researchgate.net/publication/258813772_Retail_Video_Analytics_An_Overview_and_Survey

Intel. 2018. python-cv-sample, Motion Heatmap. Luettu 3.4.2019. https://github.com/intel-iot-devkit/python-cv-samples/tree/master/examples/motion-heatmap

Adrian Rosebrock. 2018. OpenCV People Counter. Julkaistu 13.8.2018. Luettu 3.4.2019 https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/

LIITTEET

Liite 1. OpenCV-tiheyskartta


```
'''
Copyright (c) 2017 Intel Corporation.
Licensed under the MIT license. See LICENSE file in the project root
for full license information.

Edited by Pasi Rintamäki & Miiro Opas 2019 under MIT license
'''

import numpy as np
import cv2
import copy
from collections import deque

def main():
    cap = cv2.VideoCapture('students003.avi')
    num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    """
    Parameters for heatmap generation
    step = Frameskip
    tresh = Binary treshold for setting the result to maxValue
    maxValue = amount of motion to be picked up 1 = least and bigger
value means more motion captured
    colormap = OpenCV colormap to use JET is simple and goodlooking
HOT is also usable but less clear
    blur = Ammount of blur in heatmap. Makes it easier to look at
    """
    step = 2
    thresh = 2
    maxValue = 2
    colormap = cv2.COLORMAP_JET
    blur = 1

    first_iteration_indicator = 1
    for i in range(0, num_frames, step):
        print(num_frames - i)
        '''
        There are some important reasons this if statement exists:
            -in the first run there is no previous frame, so this ac-
counts for that
            -the first frame is saved to be used for the overlay after
the accumulation has occurred
            -the height and width of the video are used to create an
empty image for accumulation (accum_image)
        '''
        if first_iteration_indicator == 1:
            fgbg = cv2.bgsegm.createBackgroundSubtractorMOG()

            ret, frame = cap.read()
            first_frame = copy.deepcopy(frame)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            height, width = gray.shape[:2]
            accum_image = np.zeros((height, width), np.uint8)
```

(jatkuu)

```
            blanck = accum_image
            first = np.zeros((height, width), np.uint8)
            first_iteration_indicator = 0

        else:

            ret, frame = cap.read()  # read a frame
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # convert
to grayscale

            fgmask = fgbg.apply(gray)  # remove the background

            # apply a binary threshold only keeping pixels above
thresh and setting the result to maxValue.  If you want
            # motion to be picked up more, increase the value of
maxValue.  To pick up the least amount of motion over time, set
maxValue = 1
            ret, th1 = cv2.threshold(fgmask, thresh, maxValue,
cv2.THRESH_BINARY)

            # add to the accumulated image
            accum_image = cv2.add(accum_image, th1)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # apply a color map
    color_image = im_color = cv2.applyColorMap(accum_image, colormap)
    blurred_color_image = cv2.blur(color_image, (blur, blur))
    cv2.imwrite('blur.jpg', blurred_color_image)

    # overlay the color mapped image to the first frame
    result_overlay = cv2.addWeighted(first_frame, 0.7,
blurred_color_image, 0.7, 0)

    # save the final overlay image
    cv2.imwrite('diff-overlay.jpg', result_overlay)

    # cleanup
    cap.release()
    cv2.destroyAllWindows()


if __name__=='__main__':
  main()
```