Miika Pernu

Grid-EYE -ihmislaskuri

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

29.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Miika Pernu Grid-EYE -ihmislaskuri 21 sivua 29.5.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Lehtori Ilpo Kuivanen Teknologiajohtaja Petri Hänninen

Insinöörityössä tarkoituksena oli luoda Grid-EYE -anturiin ja sen tuottamaan lämpökuvaan pohjautuva ihmislaskurisovelluksen prototyyppi. Työ oli tarkoitus toteuttaa käyttäen tilaajan ennalta valitsemia komponentteja. Sovelluksen tavoitteena oli osoittaa konseptin toimivuus sekä luoda toimiva prototyyppilaite, jonka pohjalta jatkaa konseptin tuotteistamista.

Työ sisälsi ihmisten seurannassa alustavasti käytettävän algoritmin sekä algoritmia tukevien ohjelmallisten komponenttien luomisen. Tämän lisäksi valmis ohjelmakoodi tuli optimoida resurssien käytön suhteen ja implementoida STM32 Nucleo-L0 -sarjan kehitysalustalle.

Insinöörityön lopputulos oli pääpiirteittäin suunnitellun mukainen ohikulkevia ihmisiä suunnan ja määrän mukaan laskeva mikro-ohjaimelle implementoitu sovellus. Prototyyppilaite saatiin kasaan ja algoritmien toiminta todettiin alustavien testien valossa riittäväksi. Lisäksi prototyypin käyttämä tietoliikenneyhteys saatiin implementoitua ja testattua. Tätä kautta konsepti lopulta osoitettiin toimivaksi ja jatkokehittelyn arvoiseksi.

Avainsanat	Grid-EYE, esineiden internet, ihmislaskuri, anturointi, anturisovellus

Author Title Number of Pages Date	Miika Pernu Grid-EYE -people counter 21 pages 29 May 2018
Degree	Bachelor of Engineering
Degree Programme	information and communications technology
Professional Major	Software engineering
Instructors	Ilpo Kuivanen, Senior Lecturer Petri Hänninen, Head of Technology

The goal of the thesis was to develop an experimental people counter like application using thermal imaging of Grid-EYE -sensor. The project was intended to be built using electronical components preselected by the employer. The purpose of the prototype application is to function as a proof of concept ultimately with productization in mind.

The thesis included development of a prototype tracking algorithm and it's auxiliary software components. In addition the resulting program code was to be optimized in terms of resource usage and to be implemented into a STM32 Nucleo-L0-series development board.

The result of the project was a functioning prototype capable of measuring the number and direction of passing people in the area of measurement. The physical prototype device was built and the performance of written algorithms was deemed sufficient. Additionally the planned data connection was implemented and it's function was tested with positive results. Finally, the result was a proof of concept and in the light of result further development seems reasonable.

Keywords	Grid-EYE, internet of things, people counter, sensoring, sensor application

Sisällys

1 Johdanto

Insinöörityön tavoitteena on kehittää yrityksen aikaisemman tutkimustyön pohjalta ns. ”ihmislaskuria”, jolla voidaan seurata kohdealueella kulkevian ihmisten määrää ja kulkusuuntaa. Insinöörityön tilaaja Small Data Garden Oy on vuonna 2017 perustettu tietoliikennesovelluksiin sekä niihin liittyviin anturointi- ja mittauksiin erikoistuva yritys (1).

Yrityksen toiveena insinöörityössä on kartoittaa jo ennalta valitun laitteiston soveltuvuutta ihmislaskurin kaltaiseen sovellukseen. Lopullisena ajatuksena on kehittää laitteen loppukäyttäjän näkökulmasta yksinkertainen tuote, joka kerää ja analysoi dataa ihmisten liikkeistä kohdealueella sekä automaattisesti välittää prosessoidun datan geneeriseen pilvipalveluun. Yritys on todennut edellä mainitulle tuotteelle olevan merkittävää kysyntää.

Tilaaja on alustavasti tutkinut työn aihetta ja valinnut projektia varten käytettävän laitteiston. Ihmisten havaitsemiseen käytettävä anturi on Panasonicin Grid-EYE® Infrared Array Sensor, jonka tuottamasta lämpökuvasta havaitaan ja seurataan ihmisiä. Anturilta kerätty data on tarkoitus prosessoida paikallisesti STMicroelectronicsin STM32-tuoteperheen mikro-ohjaimella, jotta se voidaan lähettää eteenpäin geneerisellä tietoliikenneyhteydellä. Varsinainen malli valitaan tarkempien ohjelmiston asettamien vaatimusten perusteella.

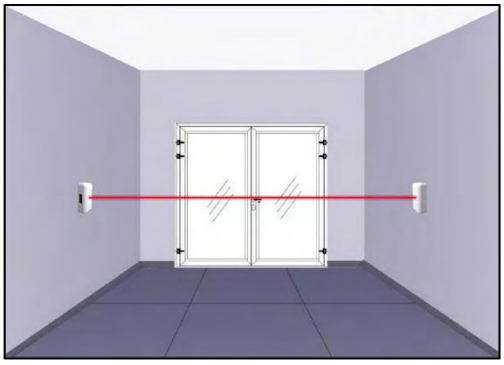
Tilaajan tekemän alustavan tutkimustyön tuloksena on kevyt ohjelma, joka käsittelee anturin tuottamat arvot ja muodostaa arvojen pohjalta binäärikuvan, josta on selkeästi erotettavissa ihmishahmot. Vastaavasti laitteisto löytyy käyttövalmiina projektin jatkoa varten. Lopulta työ rajaantuu tilaajan ohjelman tuottamien binäärikuvissa näkyvien hahmojen liikkeiden seurannan luomiseen sekä kaiken ohjelmakoodin kääntämiseen C/C++-kielellä sekä implementoimiseen mikro-ohjaimen tasolle.

2 Ihmislaskurit

Ihmislaskureita käytetään useissa eri kohteissa ja käytölle löytyy monia motiiveja. Seuraamalla ja laskemalla ihmisten määrää ja kulkureittejä voidaan ennustaa asiakaskäyttäytymistä. Tyypillisiä käyttökohteita on esimerkiksi tavaratalot, museot tai lentokentät. Ennustetun asiakaskäyttäytymisen avulla voidaan suunnitella ja optimoida kunkin kohteen toimintaa. Tavaratalossa dataa voidaan käyttää optimoimaan tuotesijoittela tai henkilökunnan sijaintia rakennuksessa. Museoissa puolestaan voidaan seurata, mikä esitys kerää eniten väkeä ja mikä vähiten. (2; 3.)

Erilaisia ihmisten seurantaan ja laskentaan tarkoittettuja sovelluksia löytyy paljon sekä harrastelija- että kaupalliseen käyttöön. Modernit sovellukset perustuvat useisiin erilaisiin teknologioihin kuten esimerkiksi konenäköön, yksinkertaisiin infrapunasädeliiketunnistimiin ja lämpökameroihin. Kullakin teknologialla on omat vahvuutensa ja heikkoutensa. Laitteen hinta vaihtelee rajusti riippuen käytetystä teknologista.

Yksinkertaiseen infrapunasäteeseen perustuva ihmislaskuri tyypillisesti asennetaan monitoroitavaan alueeseen johtavan käytävän tai oven läheisyyteen. Kaksiosainen laskuri koostuu infrapunasäteen lähettävästä sekä vastaanottavasta komponentista. Komponentit asetetaan vastakkain kuten kuvassa 1 on nähtävillä. Laskuri perustuu infrapunasäteen vastaanottimen havaitsemiin katkoihin säteessä, joita aiheuttaa esimerkiksi komponenttien välistä kulkeva ihminen tai esine. (4.)



Kuva 1. Havainnollistava kuva infrapunasäteeseen perustuvasta ihmislaskurista asennuksen jälkeen. (4.)

Tyypillisesti infrapunasäteeseen perustuvan laskurin etuna on laitteen yksinkertaisuudesta johtuva suhteellisen halpa hinta sekä laitteen helppo asennus (5). Infrapunasäteen katkeaminen tuottaa hyvin yksinkertaista dataa, jonka pohjalta tehty seuranta on vastaavasti tarkkuudeltaan vain suuntaa antavaa ja epätarkkuus korostuu komponenttien välisen etäisyyden laajetessa (6, s. 3.)

Konenäköön perustuvia ihmislaskuri etsii ihmisiä tavallisesta videokuvasta sekä seuraa niiden liikkeitä (6, s. 6). Tavanomaisesti videokuvaa analysoiva laskuri asennetaan seurattavan alueen yläpuolelle, jolloin ihmiset ovat helpommin tunnistettavissa ihmisryhmistä verrattaen tilanteeseen, jossa videokuva tulee sivusta ja rinnakkain kävelevät ihmiset voivat jäädä toistensa taakse piiloon.

Konenäköön perustuva ihmislaskuri saavuttaa oikeissa olosuhteissa helposti merkittävän tarkkuuden (<90%) (7). Videokuvan pohjalta voidaan karkeasti tunnistaa ihmisten lisäksi ihmisten ikää (lapsi vs. aikuinen) tai mahdollisesti ihmisten mukana kulkevaa tavaraa (6, s. 6).

Kuten Ankit Sachan esittelee artikkelissaan ”Embedded Computer Vision: Which device should you choose?” konenäköä voidaan soveltaa useilla eri yhden piirilevyn tietokone-alustoilla (8). Laitevaatimukset ovat huomattavasti suuremmat kuin esimerkiksi infrapunasäteeseen perustuvan laskurissa, joka on rakennettu Atmega328-mikro-ohjaimelle (9). Korkean hinnan lisäksi konenäköön perustuva ihmislaskuri saattaa olla altis esimerkiksi valon vaihtelulle seurattavassa ympäristössä. (6, s. 6.)

Lämpökuvaan pohjautuvissa järjestelmissä seurataan ihmisten liikkeitä mittaamalla ja tutkimalla kohdealueen lämpötilavaihtelua (6, s. 7). Konenäköä soveltavien järjestelmien tapaan lämpökuvaa tuottava anturi asennetaan yleensä seurattavan kohdealueen yläpuolelle samoista syistä kuin konenäköä soveltavan laitteen kamera.

Lämpökuvien etuna on suhteellinen yksinkertainen ohjelmistokoodi. Ohjelmisto tunnistaa ihmiset liikkuvina taustasta poikkeavina lämpöarvojen klustereina. Tyypillisesti järjestelmä kykenee saavuttamaan myös merkittävän tarkkuuden laskennassa. Ongelmiksi kuitenkin muodostuu esimerkiksi lapsien ja esineiden erottaminen kuvasta, ja laite on herkkä ympäristön äkilliselle lämpötilan muutoksille. (6, s. 7.)

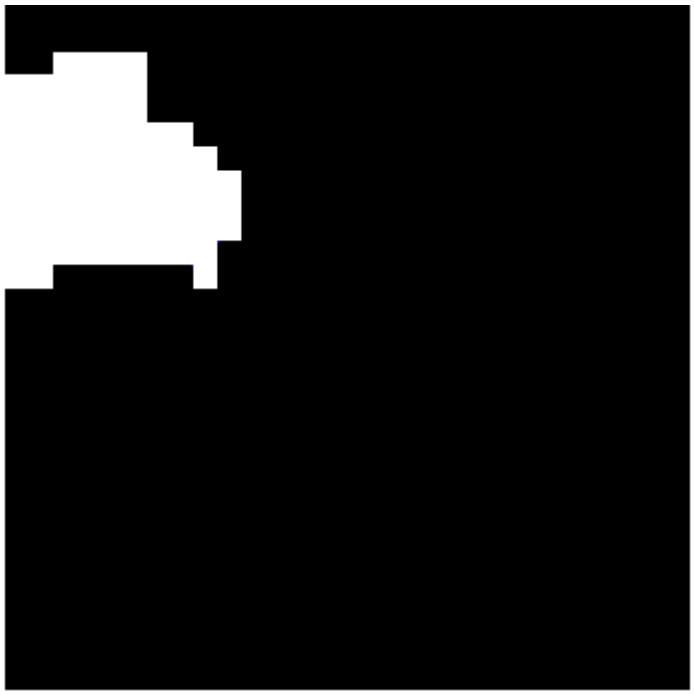
3 Projektin avainkomponentit

Suoraan henkilön liikkeen seurantaan sidonnaiset ohjelmistokomponentit jakautuvat karkeasti kolmeen osaan, jotka ovat kuvan muodostaminen anturin arvoista sekä kuvan käsittely, muotojen tunnistaminen kuvasta sekä kuvien välisten erojen eli liikkeen tunnistaminen. Kukin komponenteista on projektin onnistumisen kannalta äärimmäisen kriittinen. Kuva on käsiteltävä sellaiseen muotoon, että sieltä saadaan suodatettua merkityksettömät arvot ulos, jotta kuvasta voidaan tunnistaa ihminen vaivatta. Kuvasta taas puolestaan on löydettävä ihminen, jotta ihmisen liikettä voidaan seurata kuvasarjasta.

Anturiksi valikoitunut Panasonicin Grid-EYE® Infrared Array Sensor (AMG8833) tuottaa 8x8 (64 pikselin) matriisin kohdealueen lämpötiloista. Anturin mittaustaajuus on korkeintaan 10 Hz ja virrankulutus normaalissa käytössä 4,5 mA. Anturin virhemarginaali on ± 2,5 °C. Valmistajan mukaan anturin mahdollisiin sovelluskohteisiin lukeutuvat mukaan ihmislaskurit (10; 11).

Laitteen tietoliikenneyhteys toteutetaan Sigfox-moduuli Wisol SFM10R1:llä. Kyseessä on LPWAN (low-power wide area network) eli pienitehoinen laajaverkko. Sigfox toimii Euroopassa 868 Mhz:n taajuudella. Sigfox-yhteyttä pitkin lähetetyt viestit ovat kooltaan hyvin rajallisia, ja kuhunkin viestiin mahtuu ainoastaan 12 tavua dataa, mikä tarkoittaa, että lähetettävä data on valikoitava ja pakattava tarkasti ennen lähetystä. SFM10R1:tä ohjataan UART-yhteyttä pitkin lähetettävillä AT-komennoilla. (12; 13.)

Tilaajan tekemässä alustavassa työssä anturin tuottamasta matriisista luodaan korkeampiresoluutioinen binäärikuva. Matriiseja haetaan n (10 > n > 1) kappaletta ja niitä interpoloimalla muodostetaan yksi 32x32 resoluutioinen binäärikuva. Ohjelman käynnistyessä haetaan kymmenen mittauksen sarja kohdealueelta, joka käsitellään vastaavalla tavalla ja määritetään taustaksi. Vähentämällä tausta kustakin uudesta kuvasta jäljelle jää ainoastaan ihmisten muodostamat hahmot (ks. kuva 2).



Kuva 2. Ohjelmiston tuottama 32x32-binäärikuva. Kuvassa ihminen ylhäältä päin kuvattuna ja havaittavissa klusterina valkoisia pikseleitä kuvan vasemmassa yläreunassa.

Alustava työ on tehty GNU Octavella, joka on matematiikkaan, erilaisiin laskutoimituksiin, algoritmien suunnitteluun, datan analyysiin ja visualisointiin erikoistuva ohjelmisto. Octave muistuttaa Matlabia hyvin vahvasti sekä kielellisesti että rakenteellisesti. Samankaltaisuudet yltävät siihen asti, että Matlabia käyttäen kirjoitettu ohjelmakoodi voi toimia Octavessa sellaisenaan täysin ilman käännöstyötä (14). Erilaisista ohjelman kirjastoista löytyy monia tunnettuja algoritmejä useisiin eri tarkoituksiin mukaanlukien projektissa käytettävien binäärikuvien käsittelyyn (15). Octave on vapaan lähdekoodin ohjelmisto eikä siitä tarvitse maksaa minkäänlaisia lisenssimaksuja. (16.)

4 Liikkeen seurannan toteuttaminen

Aloittaessani insinöörityön tutustuin valmiiseen koodiin Octave-ympäristössä. Octave oli minulle täysin tuntematon projektin alussa ja tiesin paremmin tunnetun Matlabinkin vain nimellisesti. Tutustuin Octaven mahdollisiin vaihtoehtoihin ja löysin kaksi vastaavaa ohjelmistoa: Scilabin ja Sagen. Kumpikin vaihtoehdoista vaikutti lupaavilta mutta tarvetta ympäristön vaihtoon ei kuitenkaan missään vaiheessa ilmennyt. Lisäksi

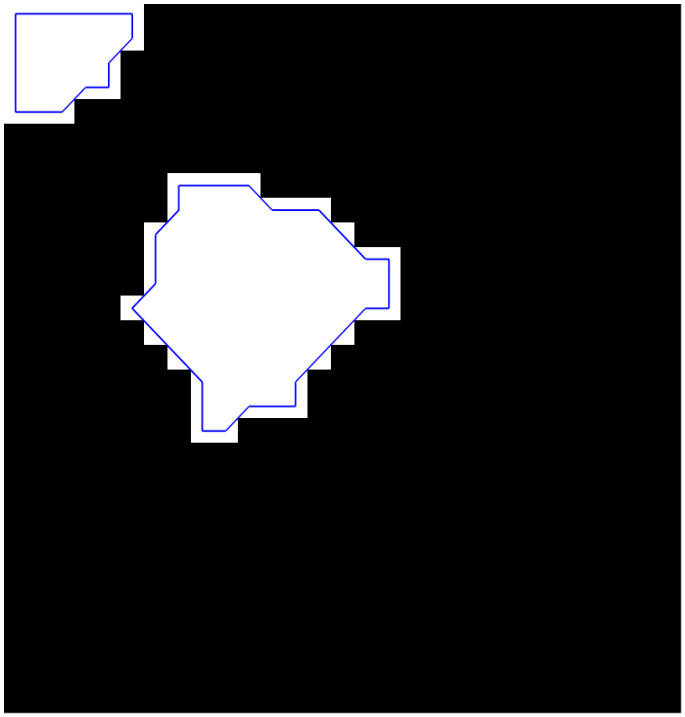
Matlab on laajalti käytetty, ja korkea yhteensopivuus Matlabin kanssa tarkoittaa, että verkosta löytyy paljon tukea koodin kirjoittamiseen.

Ihmislaskuri on lopullisesti tarkoitus toteuttaa mikro-ohjaimelle, ja ohjelmistokoodi tulisi kirjoittaa C/C++-kielellä. Projektin ohjelmallisten komponenttien kasaaminen ja testaus korkeamman tason kielellä Octave-ympäristössä vaikutti mielestäni kuitenkin parhaalta lähestymistavalta. Kirjoittamalla ohjelmakoodin ensimmäiseksi täysin Octavelle pystyin keskittymään täysin ohjelman logiikan toteuttamiseen. Vastaavasti mikro-ohjaimelle suoraan koodia kirjoittaessani olisin joutunut alusta alkaen kiinnittämään paljon huomiota koodin optimointiin, jotta se ylipäätänsä toimisi PC-ympäristöä massiivisesti pienitehoisemmalla mikro-ohjaimella.

Ensimmäisenä ongelmana projektissa oli muotojen tunnistaminen sekä sijainnin määrittäminen käsiteltävästä binäärikuvasta. Kuten kuvassa 2 on nähtävillä, ihminen näkyy kuvassa kiinteänä klusterina valkoisia pikseleitä. Alustavasti tilaaja ehdotti klustereiden etsimistä K-means-algoritmilla, jota tilaaja oli alustavasti ehtinyt jo testata. K-means on valvomattoman koneoppimisen algoritmi, joka etsii datasta ennalta määrätyn lukumäärän klustereita eli mahdollisesti samaan kategoriaan kuuluvia datapisteitä. Kutakin klusteria edustaa painopiste, jota voitaisiin käyttää ilmaisemaan klusterin tarkkaa sijaintia kuvassa. Painopiste on käytännössä klustereiden pisteistä laskettu keskiarvo. (17.)

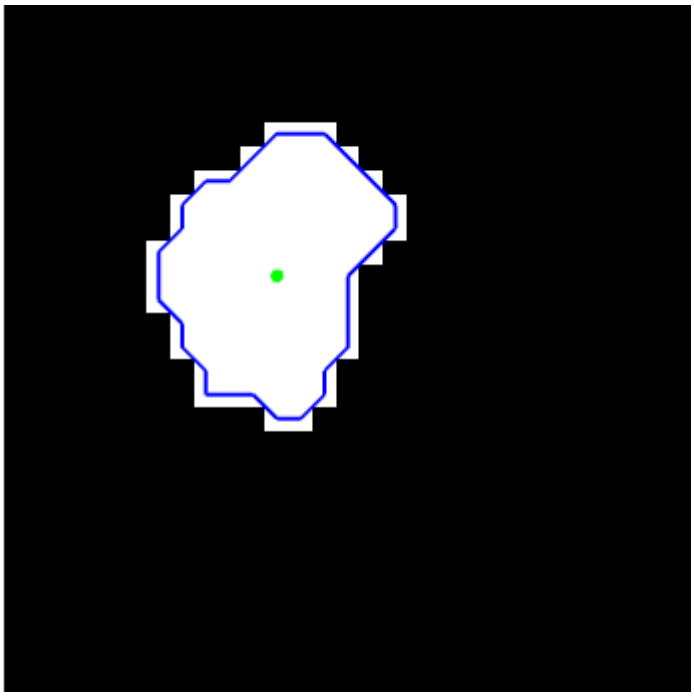
K-means-algoritmin käyttö osoittautui ongelmalliseksi. Vaatimus klustereiden lukumäärän määrittämisestä ennen algoritmin ajoa tarkoittaa, että K-means tarvitsisi tuekseen ainakin yhden algoritmin etsimään luotettavasti yhtenäisten klustereiden lukumäärän. Lisäksi ihmisten seisoessa lähellä toisiaan kuvan klusterit osuvat hyvin lähelle toisiaan. Tällaisessa tilanteessa K-means saattaa toisinaan arvioida klusterit virheellisesti. Virheellinen arvio ilmenee esimerkiksi yhden kiinteän pikseliryhmän jakautumisena kahdeksi.

Luotettavimmaksi algoritmiksi klustereiden tunnistamiseen kuvasta löytyi Octaven image-kirjastosta. Algoritmi etsii binäärikuvasta klusterille ulkoreunat ja palauttaa listan reunan pisteistä. Algoritmi yksinkertaisesti käy läpi vierekkäisiä samankaltaisia arvoja, eikä se loogisesti kykene tuottamaan samankaltaista virhettä kuin K-means. (18.) Kuvassa 3 on visualisoitu algoritmin tuottama tulos.



Kuva 3. Kuvassa kaksi henkilöä, joiden lämpöjälkien ulkorajat on tunnistettu ja visualisoitu sinisellä.

K-means-algoritmin laskemaa painopistettä olisi voitu sellaisenaan käyttää ilmaisemaan klusterin sijaintia kuvassa. Klusterien reunat etsivät algoritmi tarvitsee puolestaan tuekseen toisen erillisen algoritmin laskemaan klusterin painopisteen klusterin tarkan sijainnin määrittämiseksi kuvassa. Muotojen keskipisteiden laskemiseen käytin Octaven geometry-kirjastosta löytyvää algoritmia, joka laskee keskipisteen kaksiulotteisten pisteiden sarjasta. Laskettu ja visualisoitu keskipiste on demonstroitu kuvassa 4.



Kuva 4. Kuvassa on laskettu keskipiste henkilön lämpöjäljelle. Keskipiste on merkitty vihreällä.

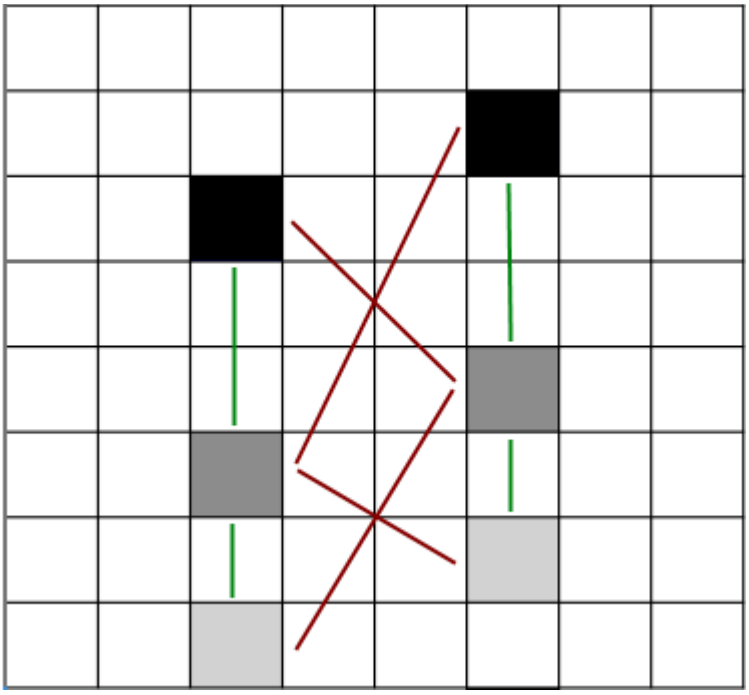
Kun kuva on käsitelty ja sieltä on haettu vaadittava data eli klustereiden pinta-ala ja sijainti, voidaan tietojen perusteella ruveta selvittämään liikkeitä kuvien sarjasta. Päädyin hakemaan liikettä kuvien välillä kahden keskeisen olettamuksen perusteella: ihminen liikkuu rajallisella nopeudella ja ihmisen liike kohdealueella on lähes aina tarkoituksenmukaista.

Ihmisen liikkeen rajallinen nopeus tulee ilmi siten, että kohdealueella liikkuva henkilö ei kykene normaalioloissa loikkimaan siten, että hän esiintyisi peräkkäisissä kuvissa satunnaisesti täysin eri puolella kuvaa. Sen sijaan liike taltioituu useaan lämpökuvaan matkan varrella pisteestä A pisteeseen B. Vastaavasti ihmisen liike kohdealueella ei ole satunnaista. Mikäli laite on asennettu käytävälle tai oven tuntumaan, ihmisen motiivina on lähes poikkeuksetta kulkea tiettyyn suuntaan mittausalueen läpi. Siten voidaan olettaa, että on epätodennäköistä, että ihmisen liikevektorin suuntaan tulee äkillisiä tai suuria muutoksia anturin mittausalueella.

Ihmisen liikkeitä voidaan seurata käytännössä vertailemalla kuvaa ajanhetkeltä t_0 suhteessa kuvaan ajanhetkeltä $t_{-1.}$ Kuvasta t_0 etsittyjen klustereiden sijainteja verrataan kuhunkin kuvan t_{-1} klustereiden sijainneista. Tämänlaisella vertailulla saadaan haettua kaikki mahdolliset klustereiden siirtymät, joista valitaan todennäköisin oikeaksi.

Yhden henkilön seuraaminen kohdealueella on jokseenkin vaivatonta, sillä kussakin kuvassa esiintyy yksi klusteri eikä merkittävää päätöksentekoa tarvitse tehdä. Klusteri liikkuu johdonmukaisesti ensimmäisestä kuvasta viimeiseen kuvaan yhdeltä reunalta kohti toista kulkiessaan kohdealueen lävitse. Tällaisessa tilanteessa ihmisen kulkeman polun määrittäminen on helppoa, sillä suurella todennäköisyydellä kukin klustereista esittää samaa henkilöä ja kussakin iteraatiossa voidaan todeta, että mahdollisia kuvien välisten klustereiden siirtymiä on vain ja ainoastaan yksi. Tilanne monimutkaistuu merkittävästi kohdealueen ihmismäärän kasvaessa.

Kun kuvassa liikkuu useita ihmisiä samanaikaisesti, todennäköisin siirtymä valitaan edellä mainittujen olettamuksien perusteella. Ensimmäisenä mahdollisista siirtymistä karsitaan etäisyyden perusteella epätodennäköisimmät vaihtoehdot kokonaan pois. Mikäli siirtymä on merkittävästi odotettua pitempi, se voidaan muitta mutkitta suoraan hylätä mahdottomana. Tämän jälkeen verrataan edellisen iteraation eli kuvien t_{-1} ja t_{-2} välisiä hyväksyttyjä siirtymiä liikevektoreiden suunnan perusteella tuoreisiin jäljellä oleviin mahdollisiin siirtymiin ja katsotaan, mikä siirtymistä vastaa parhaiten suunnaltaan edellisiä.



Kuva 5. Seuranta-algoritmin toiminta havainnollistettuna. Mustalla merkitty kuvan t_0-klusterit, tumman harmaalla kuvan t_{-1}-klusterit ja vaalean harmaalla t_{-2} klusterit. Toteutuneet siirtymät merkitty vihreällä ja vaihtoehtoiset mahdolliset siirtymät punaisella.

Useiden iteraatioiden jälkeen algoritmi muodostaa sarjan siirtymiä. Sarjasta on osattava kuitenkin erikseen vielä päättää, milloin se voidaan määrittää ohikulkeneeksi ihmiseksi. Käytännössä päätöstä voidaan tehdä muutamalla eri kriteerillä, joihin lukeutuu sarjan elinkaaren pituus, sarjan ensimmäisen ja viimeisen pisteen nettosiirtymä akselilla sekä sarjan päätyminen lähelle kohdealueen reunoja.

Abstraktilla tasolla kutakin siirtymää voidaan ajatella ketjun linkkinä, joka koostuu kahdesta pisteestä. Pisteinä on siirtymän alkupiste kuvasta t_{-1} ja loppupiste kuvasta t_0. Kustakin iteraatiosta saadaan klustereiden määrän mukainen määrä myöskin linkkejä, jotka voidaan yhdistää edellisen iteraation linkkeihin alku- ja loppupisteiden perusteella. Usean iteraation jälkeen linkeistä on muodostunut ketju, joka esittää ihmisen kulkemaa reittiä. Lopulta ketjuista lasketaan ohi kulkeneet ihmiset suunnan mukaan, ja valmis prosessoitu data on lähetettävissä pilvipalveluun.

5 Implementaatio mikro-ohjaimelle

Ohjelmakoodin prototyypin valmistuttua Octavelle, aloitin ihmislaskurin kehitystä mikro-ohjaimelle, mikä on projektin ehdottomasti haastavin osuus. Implementaation tavoitteena on saada kaikki ihmislaskurissa käytettävä ohjelmakoodi käännettyä C/C++-kielelle siten, että se on ajettavissa täysin paikallisesti valitulla alustalla.

Tämän vaiheen ajan pyrin systemaattisesti kääntämään ohjelmakoodia loogisessa järjestyksessä Octavelle kirjoitetun prototyypin mukaisesti. Teen käännöksiä funktio kerrallaan kohdekielelle, jonka jälkeen pyrin testaamaan ja todentamaan funktion toiminnan. Käännöstyön aikana voin visualisoida ja tutkia funktioiden toimintaa helposti viemällä dataa UART-väylää pitkin kehitysalustalta PC-ympäristöön Octavelle.

Tilaaja oli valinnut käytettäväksi mikro-ohjaimeksi STM32 Nucleo-L031K6 -kehitysalustan. Kyseessä on STMicroelectronics-yrityksen valmistama L0-sarjaan kuuluva mikro-ohjain, joka on kehitetty hyvin energiatehokkaaksi (19; 20). Laitteelta kuitenkin katsottiin löytyvän riittävästi resursseja projektin toteuttamiseen. Kehitysalustan tarkempi projektin kannalta tärkeä spesifikaatio on näkyvissä taulukossa 1.

Taulukko 1. Nucleo-L031K6-kehitysalustan spesifikaatioita.

Flash-muisti	32 Kt
RAM-muisti	8 Kt
Virrankulutus ajon aikana alkaen	~50 µA
Virrankulutus lepotilassa	~340 nA
Järjestelmäkellon maksimitajuus	32 MHz
Projektin kannalta merkittäviä oheislaitteita	reaaliaikakello

Projektin alusta alkaen oli selkeää, että käännöstyön aikana on kiinnitettävä erityisesti huomiota siihen, että ohjelmakoodi pysyy yksinkertaisena sekä tehokkaana. Kuten taulukosta 1 käy ilmi, laitteella ei mahdu kovinkaan suurta ohjelmaa eikä RAM-muisti riitä tukemaan isojen datamäärien käsittelyä samanaikaisesti.

Tilaaja rohkaisi minua kirjoittamaan mahdollisimman paljon ohjelmakoodista ilman kolmannen osapuolen kirjastoja, jotta projektin tuotosta voidaan kaupallistaa ilman massiivisia muutoksia ohjelmakoodiin mahdollisten lisenssien ja käyttöoikeuksien asettamien rajoitusten takia. Vastaavasti kirjoittamalla suurimman osan ohjelmakoodista itse, pystyn varmistamaan, että resurssien käyttö pysyy hallittavissa.

Implementointivaiheen lähestyessä tilaajan edustaja kirjoitti tarvittavan ohjelmakoodin itse kehitysalustan hallinnointiin. Tämä käytännössä pitää sisällään anturin vaatiman I2C-väylän käyttöön tarvittavat funktiot, testauksessa käytettävän UART-väylän käyttöön tarvittavat funktiot sekä laitteen järjestelmäkellon taajuuden nostamisen maksimiin. Käytännössä siis pääsin aloittamaan implementaation siten, että raakadata anturilta on haettavissa.

Työvaiheet lopulta ovat käytännössä samat kuin Octave-prototyypin kohdalla. Suurena erona kuitenkin on, että joudun kirjoittamaan itse kuvankäsittelyssä käytetyt algoritmit sekä algoritmit klustereiden etsimiseen kuvasta siinä, missä nämä olivat Octave-kirjastoissa valmiina.

Pyrin kirjoittamaan ohjelmakoodia siten, että projekti pysyy helposti hallittavissa ja suunnittelu vaiheessa jaottelin toiminnallisuutta eri luokkiin. Luokkia muodostui siten, että kaikki aktiivisessa käytössä olevat osat eli anturiin ajoon liittyvä koodi, kuvankäsittelyyn liittyvät funktiot sekä varsinaiset seurannassa käytetty data ja funktiot

muodostivat omat luokkansa. Tämän lisäksi päädyin kuvaamaan luokkina myös mm. pisteitä koordinaatistolla, pisteiden välisiä linkkejä sekä linkeistä muodostuvia ketjuja.

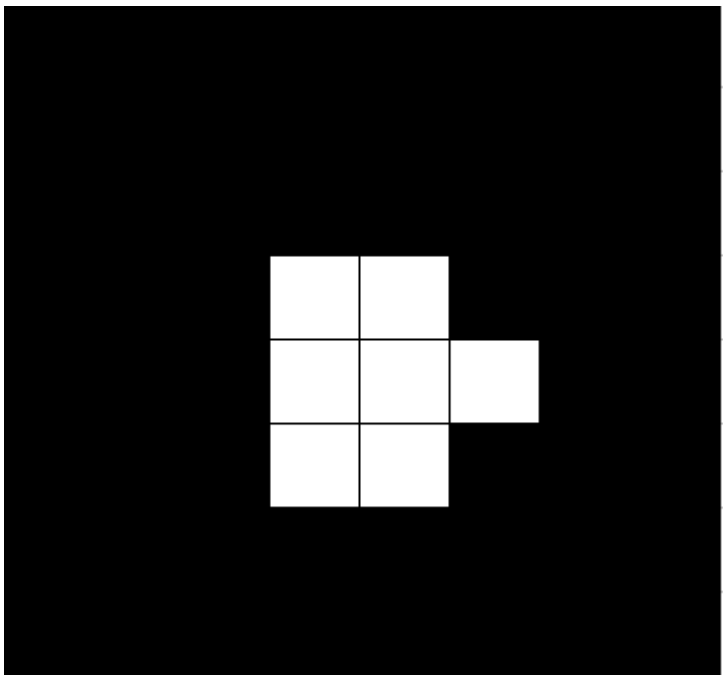
5.1 Kuvan käsittely

Binäärikuvan muodostamisessa erityisen kriittiseksi vaiheeksi kehityksen aikana osoittautui taustan määrittäminen. Mikäli taustan mittauksen aikana mittausalueella kulkee ihminen, se vaikeuttaa ihmisen tunnistamista samassa pisteessä. Tämä johtuu täysin siitä, että tässä tapauksessa ihminen katsotaan osaksi taustaa eikä seuraavissa kuvissa ohikulkeva ihminen enää samassa pisteessä erotu taustasta mitenkään. Käytännössä tämä voi pahimmassa tapauksessa johtaa siihen, että virheellisestä taustasta johtuen binäärikuvasta puuttuvan datan takia ei voida normaaliin tapaan määrittää ohikulkijan kulkemaa reittiä.

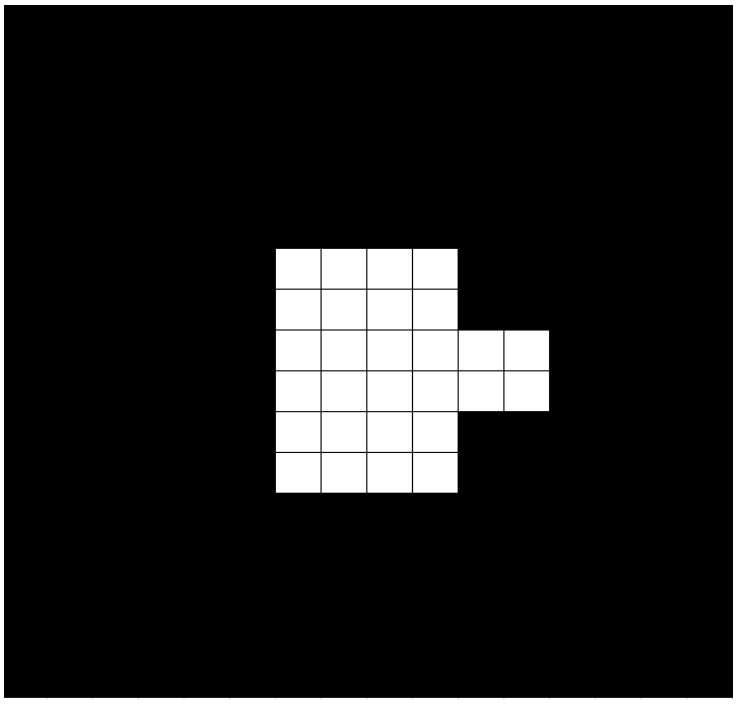
Taustan muodostamisessa voidaan käyttää apuna esimerkiksi yksinkertaista digitaalista liikkeentunnistinta, jonka mittausalue ylittää pinta-alaltaan Grid-EYE -anturin mittausalueen. Liikettä havaittaessa odotetaan ja taustalämpötila mitataan vasta liikkeen lakattua, millä voidaan korkealla todennäköisyydellä varmistaa, ettei mittausalueella ole ollut häiriötekijöitä.

Octave-prototyypissä kuvankäsittelyyn kuului usean peräkkäisen kuvan (resoluutiolla 8x8) yhdistäminen yhdeksi ja interpolointi suurempaan resoluutioon (32x32). Katsoin, että samankaltainen operaatio olisi tehtävä myös mikro-ohjain implementaatiossa, jotta kuviin saadaan hieman lisää tarkkuutta. Tutkin erilaisia interpolointimenetelmiä ja suurimpana kriteerinä on yksinkertaisuus.

Kaikkein yksinkertaisinpana ratkaisuna interpolaatioon on ns. lähimmän naapurin interpolaatio, joka käytännössä tarkoittaa, että alkuperäisestä kuvasta luodaan vain pikselitarkka suurennus. Nimensä mukaisesti interpolaatiossa suurennoksen uusien pikseleiden arvo otetaan johdonmukaisesti lähimmästä naapurista, jolloin operaatio ei ole laskennallisesti kovin raskas. (21.)



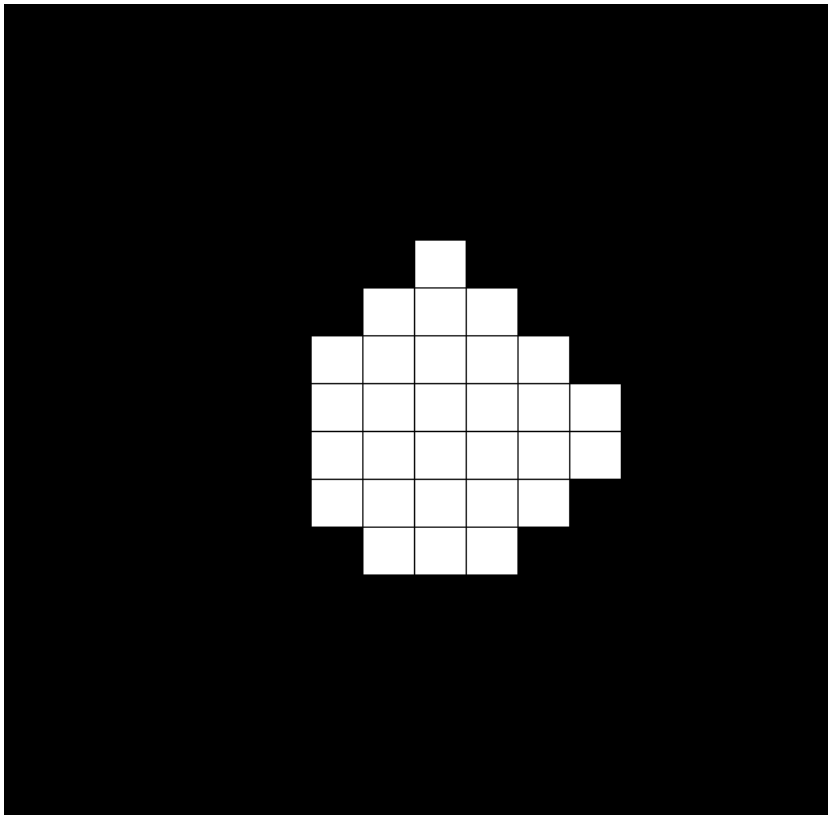
Kuva 6. Alkuperäinen interpoloimaton kuva.



Kuva 7. Kuva 6 interpoloitu käyttäen lähimmän naapurin interpolaatio-menetelmää.

Toisena yksinkertaisena vaihtoehtona on bilineaarinen interpolaatio. Bilineaarisessa interpolaatiossa uusille pikseleille johdetaan arvot laskemalla akselin lähimpien

pisteiden painotettu keskiarvo. (22). Painotettujen keskiarvojen laskeminen ei ole laskennallisesti kovin raskasta mutta huomattavasti raskaampi kuin suoraan kopioida lähimmän pikselin arvo.



Kuva 8. Kuva 6 interpoloitu käyttäen bilineaarista interpolaatio-menetelmää.

Kuten kuvista 7 ja 8 on nähtävissä, lähimmän naapurin interpolaatio ei tuo merkittävästi lisäarvoa suhteessa alkuperäiseen kuvaan vaan kyseessä on pelkkä suurennus. Bilineaarinen interpolaatio puolestaan tuottaa tarkemman oloista kuvaa, jonka muodot vastaavat tarkemmin kuvassa kulkevaa ihmistä. Vaikka bilineaarinen interpolaatio on laskennallisesti hieman raskaampi, se ei kuitenkaan vaadi merkittävällä tavalla resursseja, mikä hidastaisi huomattavasti ohjelmakoodin ajamista ja siten estäisi menetelmän käyttöä. Selväksi valinnaksi muodostui siis bilineaarinen interpolaatio.

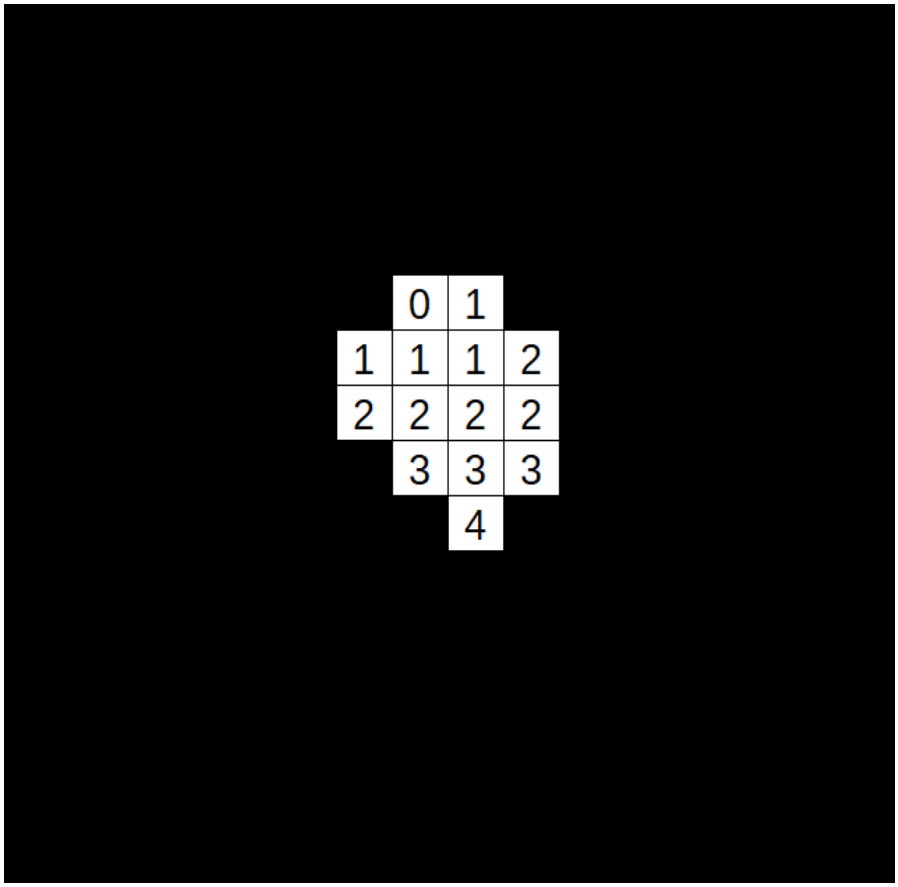
5.2 Klustereiden tunnistus ja liikkeiden seuranta

Octavelle kirjoitetussa prototyypissä klusterit eli kohdealueella olevat ihmiset tunnistettiin Octaven kirjastosta löytyvällä valmiilla algoritmilla. Mikro-

ohjainimplementaatiossa toteutus pitää kuitenkin kirjoittaa itse. Käytännössä yhtenäiset klusterit on tunnistettava siten, että niistä saadaan laskettua kaikkien pisteiden keskiarvo eli klusterin keskipiste sekä klusterin pinta-ala.

Klustereiden etsimisessä sovellan flood-fill-algoritmia. Käytännössä binäärikuvaa iteroidaan läpi pikseli kerrallaan ja arvon yksi löytyessä kutsutaan algoritmia etsimään koko löydettyyn pisteeseen liittyävä klusteri. Algoritmi tallentaa erilliseen taulukkoon kaikkien löytyneiden klustereiden keskipisteet sekä pinta-alat. Pinta-alan perusteella suodatetaan ulos todennäköisimmät kuvan virhearvot.

Algoritmi toimii käytännössä siten, että se kussakin iteraatiossa käy läpi edellisessä iteraatiossa löytyneiden pisteiden viereiset pisteet. Algoritmi lopettaa toimintansa ja palauttaa arvot, kun ensimmäinen iteraatio, jossa ei enää löydy pisteitä, tulee vastaan.



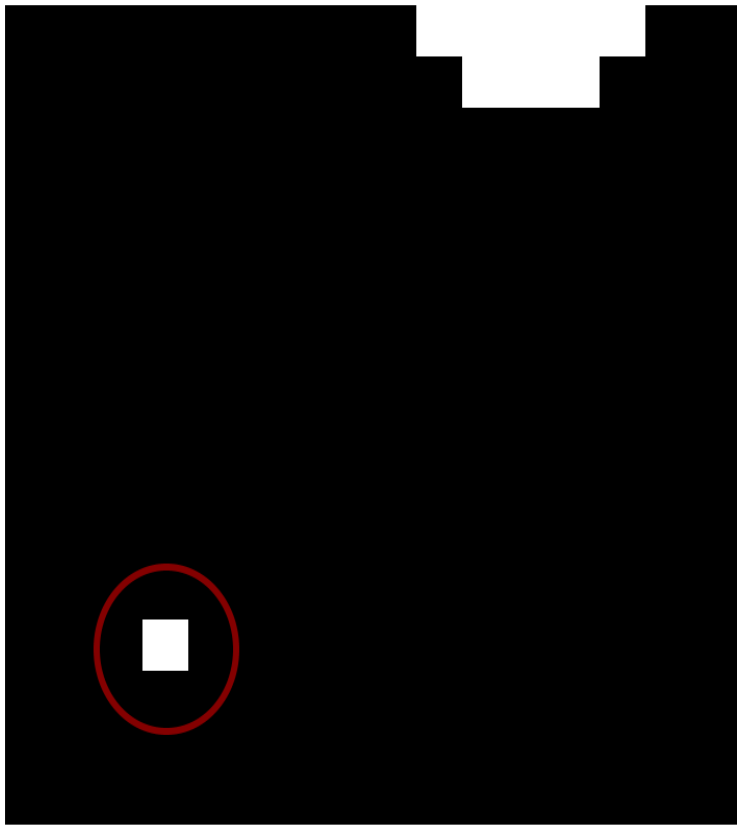
Kuva 9. Flood fill-algoritmin toimintaa havainnollistava kuva. Kunkin iteraation löytämät pisteet merkitty seuraavaa iteraatiota vastaavalla järjestysluvulla (0-4). Pikseleiden arvoja yksi merkitään kuvassa valkoisella ja nollaa mustalla.

Kuvassa 9 on nähtävissä esimerkkitapaus algoritmin toiminnasta. Algoritmi on lähtenyt käyntiin, kun ensimmäinen pikselin arvo yksi on tullut vastaan. Tätä pistettä merkitään kuvassa nollalla. Algoritmi käy läpi kaikki suoraan ensimmäisen pisteen viereiset pisteet ja merkitsee ne muistiin seuraavan iteraation järjestysluvulla, mikäli pikselit ovat samanarvoisia kuin se itse. Tässä tapauksessa löytyneitä pikseleitä merkitään luvulla yksi. Seuraavassa iteraatiossa toistetaan sama operaatio jokaisen järjestysluku yhdellä merkityn pikselin osalta. Algoritmi päättää toimintansa neljännen iteraation loppuun, sillä neljännellä iteraatiolla ei löydy enää uusia samanarvoisia pikseleitä.

Algoritmi tallentaa koko ajan jokaisen löydetyn pikselin kohdalla paikalliseen muuttajaan löydettyjen pikseleiden kokonaismäärää. Tätä kautta klusterin pinta-ala saadaan suoraan talteen. Vastaavasti algoritmi laskee kaikkien löydettyjen pisteiden x koordinaattien ja y koordinaattien summat. Algoritmin ajon päättyessä summista ja pinta-alasta voidaan laskea keskiarvot x koordinaatille ja y koordinaatille, jotka yhdessä edustavat klusterin keskipistettä.

Aikaisemmin suoritetusta kuvankäsittelystä huolimatta kuvaan eksyy toisinaan virhearvoja. Virhearvot näkyvät kuvassa esimerkiksi virheellisinä pikselin arvoina odottamattomissa pisteissä ja pääasiassa virheellisenä positiivisena arvona. Mikäli virheellisetkin arvot laskettaisiin omiksi klustereikseen, se vaikeuttaisi myöhemmin tehtävää päätöksentekoa merkittävästi, sillä se kasvattaisi mahdollisten siirtymien määrää kuvien välillä. Virhearvon päätyessä osaksi isompaa klusteria ne on vaikea suodattaa ulos. Tässä tapauksessa virhearvot voivat kuitenkin pahimmillaan siirtää klusterin keskipistettä vain yhden pikselin verran sattumanvaraiseen suuntaan. Tämä ei kuitenkaan aiheuta merkittäviä ongelmia myöhemmissä käsittelyvaiheissa.

Klustereita etsivä funktio suodattaa virhearvojen muodostamia todella pieniä klustereita pois pinta-alan perusteella. Laitteen ollessa asennettuna 2,5-3,0 m korkeuteen ihminen näkyy käsitellyssä binäärikuvassa poikkeuksetta klusterina, jonka pinta-ala on suurempi kuin 5 pikseliä, mikä tarkoittaa, että virhearvon muodostama klusteri on hyvin helppo erottaa ihmisen muodostamasta klusterista. Kuvasta kymmenen löytyy esimerkki virheellisestä arvosta. Algoritmin toiminnassa käytännössä mikä tahansa yksinäinen pikseli arvolla yksi katsotaan virhearvoksi eikä sitä koskaan merkitä kokonaisena klusterina tai käsitellä myöhemmissä vaiheissa.



Kuva 10. Esimerkki muodostuneesta virhearvosta. Kuva on otettu reaaliaikaisesta videokuvasta laitteelta, kun kuvankäsittely on jo suoritettu. Kuvassa henkilö on astumassa kuvaan oikeasta yläreunasta. Virhearvo on ympäröity punaisella. Virhearvo poistui seuraavasta kuvasta täysin.

Virhearvoille tyypillistä on se, että ne esiintyvät satunnaisesti missä tahansa kuvassa ja voivat sijaita missä tahansa pisteessä. Ne ovat kuitenkin harvoja poikkeuksia lukuunottamatta hetkellisiä ja eivätkä todennäköisesti esiinny seuraavassa kuvassa enää lainkaan samassa paikassa. Taustalämpötilalla on iso vaikutus virhearvojen esiintymisen todennäköisyyteen. Mitä lähempänä ihmisen ihon pintalämpötilaa taustalämpö on, sitä todennäköisempää on, että binäärikuvan muodostamisessa taustalämmösta johtuvia arvoja päätyy osaksi kuvaa.

Ihmisten seurantaan käytetyt algoritmit puolestaan voitiin implementoida käännöstyön jälkeen ilman merkittäviä muutoksia toiminnallisuuteen tai operaation logiikkaan, sillä kirjoitin algoritmin alunperinkin täysin itse. Ainoana merkittävänä erona on, että ketjuille ja linkeille oli asetettava maksimimäärät, ettei laitteelle tule muistinkäyttöön liittyviä ongelmia.

Ihmisten seurannan tulokset tallenetaan laskuriin, jota esitetään omana luokkanaan. Laskuri yksinkertaisesti pitää sisällään, montako ihmistä on kulkenut mihinkin suuntaan sekä, miltä ajalta mittaus on. Sigfox-viestiä koostaessa voidaan yksinkertaisesti hakea laskurin sisältö viestiin ja resetoida laskuri seuraavaa lähetystä varten.

5.3 Muita ominaisuuksia

Laitteen toiminnan kannalta on kriittistä, että itse seurantaan liittyvät ohjelmalliset ja laitteiston komponentit toimivat. Pyrin kuitenkin sisällyttämään laitteeseen muutakin toiminnallisuutta, jotta laitteen toimintakyky kokonaisuutena paranee. Erityisesti virran säästöön liittyvät ominaisuudet ovat kriittisiä, sillä laitteen on tarkoitus toimia pariston varassa.

Ensimmäisenä ominaisuutena on laitteen siirtäminen virransäästötilaan ajanjaksoiksi, jolloin mittausta ei tarvita. Laitteen monista käyttökohteista useat ovat sellaisia, jotka eivät vaadi jatkuvaa mittausta kahtakymmentäneljää tuntia vuorokaudessa. Esimerkiksi kauppaan asennettu laite voitaisiin siis ohjelmoida säästämään virtaa ainakin siksi ajaksi, kun kauppa on kiinni.

Virransäästötilassa Grid-EYE -anturi sekä itse kehitysalusta asetetaan lepotilaan, jossa ainoastaan tarvittavat oheislaitteet (esim. reaaliaikakello) ja niiden vaatimat oskillaattorit pysyvät toiminnassa. Lepotilassa virrankulutus putoaa merkittävästi. Kehitysalustaan kuuluu reaaliaikakello, joka on ohjelmoitavissa lähettämään signaalin tiettyinä kellon aikoina. Reaaliaikakellon lähettämää signaalia voidaan käytännössä käyttää herättämään laite virransäästötilasta ja ilmoittamaan tarpeesta siirtyä virransäästötilaan.

Tietoliikenneyhteyden implementointi itsessään on vaivatonta. Viestit on kuitenkin ajoitettava järkevästi, sillä lähetetty data ei ole kovinkaan arvokasta ellei voida osoittaa, että liikettä on tapahtunut selkeällä aikavälillä. Suunnittelin laitteen lähettävän tietoa mahdollista ohikulkijoista esimerkiksi 15 minuutin välein, mikäli kulkijoita on ollut. Tämäkin ominaisuus vaatii avukseen reaaliaikakelloa, jolta voidaan hakea kellonaikaa esimerkiksi jokaisen ajosilmukan lopulla lähetettyjen viestien tarkaksi ajoittamiseksi.

5.4 Ohjelman toiminta kokonaisuutena

Ohjelmointivaiheessa koin, että laitteen varsinainen toiminta on helposti jaettavissa erilaisiin tiloihin. Esimerkiksi käynnistyminen on oma muusta toiminnasta selkeästi eroava sekvenssinsä. Vastaavia muita sekvenssejä on myös aktiivinen mittaus sekä vaikkapa virransäästötilaan siirtyminen ja datan lähettäminen. Täten ohjelmiston hallinta kokonaisuutena tuntuu helpoimmalta kirjoittamalla varsinainen ajosilmukka tilakoneeksi.

Tilakone on erinomainen malli kehitysvaiheessa, sillä sitä kautta on helppo seurata laitteen toimintaa ja ongelmatapauksissa paikantaa, mikä tiloista aiheuttaa ongelmia. Tämän lisäksi tilakone mallina tekee koodista selkeästi jäsenneltyä ja helpottaa ohjelmakoodin tulkitsemista.

Luettelo eri tiloista ja niiden vastuualueista:

- Laitteen käynnistymissekvenssiin kuuluu kaikkien käytettävien muuttujien alustaminen, I2C- ja UART-yhteyden konfigurointi, sekä anturin konfigurointi ja/tai herättäminen virransäästötilasta.
- Aktiiviseen mittaukseen sisältyy taustalämpötilan määrittäminen, binäärikuvien muodostaminen, kuvankäsittely sekä kohdealueen ihmisten liikkeiden seuranta. Lisäksi tässä vaiheessa tarkkaillaan, milloin seuraava viesti on lähetettävä.
- Viestin lähetykseen kuuluu ainoastaan viestin muodostaminen laskurilta haettavasta datasta sekä viestin että lähetyskomennon kirjoittaminen SFM10R1 -moduulille.
- Virransäästötilaan siirtyminen pitää sisällään tarpeettomien oheislaitteiden sammuttamisen sekä reaaliaikakellon lähettämän herätyssignaalin konfigurointi.

Tilakoneen keskiössä toimii aktiivinen mittaus. Käynnistymissekvenssiä lukuunottamatta aktiiviseen mittaukseen liittyvät toimenpiteet määrittävät siirtymisen muihin tiloihin. Aktiivisen mittauksen tilassa laite hallinnoi kaikkea dataa ihmisen liikkeisiin liittyen ja tekee päätökset jatkotoimenpiteiden suhteen – jatketaanko vielä mittausta, siirrytäänkö lepotilaan vai lähetetäänkö dataa eteenpäin.

6 Tulosten arviointi

Grid-EYE -anturi osoittautui toimivaksi anturiksi ihmisten seurantaan hyvin pienestä resoluutiosta huolimatta. Anturilla pystytään alustavien testien valossa seuraamaan

ihmisiä kohdealueella kohtuullisella tarkkuudella riittävän hyvissä olosuhteissa. Sovellus tällaisenaan ei kuitenkaan skaalaudu kovin erilaisiin toimintaympäristöihin ja vaati jatkokehitystä.

Anturi on lämpökameran luonteen vuoksi on erityisesti altis nopeille taustalämmön muutoksille. Kuten mainitsin aikaisemmin, virhearvojen esiintymisen todennäköisyys kasvaa huomattavasti, mitä korkeampi taustalämpötila on suhteessa ihmisen ihon pintalämpötilaan. Taustalämpötilan kasvaessa riittävästi ihminen ei ole enää kuvasta erotettavissa, sillä tunnistus pohjautuu täysin lämpötilaeron havaitsemiseen taustan ja ihmisen välillä. Laitteen optimaalinen toimintaympäristö olisi siis viileähkö (n. 20 °C) huone tai käytävä. Laitteen mittaustarkkuuden kannalta on tärkeää, että taustalämpötilassa ei tapahdu suuria ja nopeaita muutoksia.

Taustalämpötilaan liittyvien vaatimusten takia laite olisi parasta asentaa sisätiloihin. Sisätiloissa laitteen mittaustarkkuudelle asettavia ympäristöllisiä riskejä on esimerkiksi ulko-ovien läheisyys. Esimerkiksi riittävän lämmin tuulenpuuska voi mahdollisesti aiheuttaa vaikeuksia ihmisten tunnistuksessa mittausalueella.

Toinen merkittävä virhearvoja tuottava tekijä on ihmisten tiheys mittausalueella. Mikäli ihmiset kulkevat liian lähekkäin toisiaan, vaarana on, että ihmisten lämpöjäljet yhdistyvät kuvassa yhdeksi isoksi klusteriksi. Tätä vastaan voidaan taistella jossain määrin ohjelmallisesti, mutta riskinä on aina, että klustereiden yhdistyessä osa ihmisistä jää siten laskematta. Tästä aiheutuneet virheet ovat pääasiassa laskemattomia ihmisiä eivätkä vaikuta laitteen toimintaan kokonaisuutena.

Laitteen asennuskorkeuskin on merkittävä tulosten kannalta. Ihmisten koko kuvissa kasvaa nopeasti, jos laite asennetaan matalalle ja sitä kautta myös ihmisen suhteellinen nopeus laitteeseen nähden kasvaa ja suunnan määrittäminen vaikeutuu. Vastaavasti ihmisten tunnistaminen ja erottaminen kuvista vaikeutuu, mikäli laite asennetaan liian korkealle ja klustereiden koko pienenee häviävän pieneksi. Optimaaliseksi korkeudeksi osoittautui noin 2,5-3,0 m.

Laite on asennettava paikkaan, josta laitteen viestit ovat kuultavissa. Tietoliikenneyhteys on toteutettu radioteitse, joten laitteen toiminnan kannalta on tärkeää, että laite asetetaan alueelle, jossa ei ole kuuluvuutta häiritseviä tekijöitä. Laitteen kuuluvuus on testattava ennen varsinaista käyttöönottoa.

Laitteen rajoitteista huolimatta insinöörityön tulokset asettavat loistavat mahdollisuudet laitteen tuotteistamiselle ja jatkokehitykselle. Laitteella on äärimmäisen paljon potentiaalia, sillä Grid-EYE -anturi on hinnaltaan todella alhainen sekä virrankulutukseltaan pieni. Tämän lisäksi laitteen toiminta jo nykyisillä algoritmeillä on hyvin lupaavaa. Yksittäisiä henkilöitä tunnistetaan huomattavan korkealla todennäköisyydellä ja useammankin henkilön kulkusuunnan määrittäminen samanaikaisesti onnistuu pääsääntöisesti hyvin.

7 Jatkokehitys

Insinöörityön tavoitteena on ollut luoda prototyyppi Grid-EYE -anturia käyttävästä ihmislaskurisovelluksesta, jonka pohjalta sovellusta voidaan tuotteistaa. Laitteen jatkokehitykseen kannattaa insinöörityön tulosten valossa panostaa, sillä sovellus osoittautui konseptina hyvin toimivaksi.

Keräämällä riittävästi testidataa erilaisista ympäristöistä, laitetta toimintaa voidaan varmentaa ja parantaa testitulosten mukaan. Vastaavasti erilaisten käyttökokemuksian kautta voidaan löytää laskuriin käyttöarvoa nostavia kehitysideoita lopullista tuotetta silmällä pitäen.

Suunnittelemalla laitteelle oman piirilevyn, josta karsitaan kehitysalustan ylimääräiset oheislaitteet, laitteen virrankulutusta voidaan pudottaa merkittävästi. Lisäksi laitteen kokoa saadaan pienemmäksi ja kokonaisuutena hinnaltaan merkittävästi halvemmaksi. Näkisin, että ohjelmoimalla ja optimoimalla ohjelmakoodia laitteen energiankulutusta voidaan vähentää merkittävästi, mikä kasvattaisi laitteen käyttöikää.

Lähteet

1 Small Data Garden Oy. 2017. Verkkoaineisto. Alma Talent Oy. <https://www.kauppalehti.fi/yritykset/yritys/small+data+garden+oy/28154722>. Luettu 9.4.2018.

2 Irisys.net. Verkkoaineisto. <https://www.irisys.net/people-counting-retail> Luettu 16.4.2018.

3 Trafsys.com. Verkkoaineisto. <https://www.trafsys.com/increase-your-retail-conversion-rate-people-counting-can-help/> Luettu 16.4.2018

4 Sensource, people counter brochure. 2001. Verkkoaineisto. <http://www.sensourceinc.com/PDF/TB12N_Counter_Brochure.pdf> Luettu 15.4.2018.

5 1st generation: Infrared beam counters (2002). Verkkoaineisto. <https://en.wikipedia.org/wiki/People_counter#1st_generation:_Infrared_beam_counters_(2002)> Luettu 15.4.2018.

6 Which Type of People Counter is Best for You? 2015. Verkkoaineisto. <https://www.trafsys.com/wp-content/uploads/2015/01/2015-1-16_People-Counter-Types_ebook-1.pdf> Luettu 15.4.2018.

7 Vyacheslav Konstantinov. 2017. Research on Methods for Counting the Number of People in a Video Stream Using OpenCV. Verkkoaineisto. <https://www.apriorit.com/dev-blog/445-computer-vision-with-opencv> Luettu 15.4.2018.

8 Ankit Sachan. 2017. Embedded Computer Vision: Which device should you choose? Verkkoaineisto. <https://www.learnopencv.com/embedded-computer-vision-which-device-should-you-choose/> Luettu 15.4.2018.

9 ARDUINO-DIY LASER / IR PERSON COUNTER. 2014. Verkkoaineisto. <http://www.instructables.com/id/IR-laser-person-counter/> Luettu 15.4.2018.

10 Panasonic Grid-EYE datasheet. Verkkoaineisto. <https://industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ADI8000C53.pdf> Luettu 15.4.2018.

11 Panasonic Grid-EYE. Verkkoaineisto.
<https://na.industrial.panasonic.com/products/sensors/sensors-automotive-industrial-applications/grid-eye-infrared-array-sensor>
Luettu 15.4.2018.

12 Sigfox-module SFM10R1. Verkkoaineisto.
<https://yadom.eu/downloadable/download/sample/sample_id/188/>
Luettu 21.5.2018.

13 Sigfox. Verkkoaineisto
<https://en.wikipedia.org/wiki/Sigfox>
Luettu 21.5.2018.

14 Jason Baker. Open source alternatives. Verkkoaineisto.
<https://opensource.com/alternatives/matlab>
Luettu 16.4.2018.

15 GNU Octave image package. Verkkoaineisto.
<https://octave.sourceforge.io/image/overview.html>
Luettu 16.4.2018.

16 GNU Octave. Verkkoaineisto.
<https://www.gnu.org/software/octave/about.html>
Luettu 15.4.2018.

17 Andrea Trevino. Datascience, K-means clustering. Verkkoaineisto.
<https://www.datascience.com/blog/k-means-clustering>
Luettu 16.4.2018.

18 GNU Octave boundaries algorithm. Verkkoaineisto.
<https://octave.sourceforge.io/image/function/bwboundaries.html>
Luettu 17.4.2018.

19 STMicroelectronics, NUCLEO-L0. Verkkoaineisto
<http://www.st.com/content/ccc/resource/technical/document/data_brief/b1/d8/13/d4/b0/b7/4b/6e/DM00214578.pdf/files/DM00214578.pdf/jcr:content/translations/en.DM00214578.pdf>
Luettu 15.5.2018.

20 STM32 L0-series. Verkkoaineisto.
<http://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/brochure/a5/a6/6b/c0/0f/32/40/2e/brstm32l0.pdf/files/brstm32l0.pdf/jcr:content/translations/en.brstm32l0.pdf>
Luettu 15.5.2018.

21 Nearest neighbor interpolation. Verkkoaineisto.
<https://angeljohnsy.blogspot.com/2017/11/nearest-neighbor-interpolation.html>
Luettu 15.5.2018.

22 Bilinear interpolation. Verkkoaineisto.
<http://supercomputingblog.com/graphics/coding-bilinear-interpolation/>
Luettu 15.5.2018.