vAI:lla tuottavuutta?

Plastec Finland Oy
Tekoäly tuotevirheiden tunnistuksessa



4.8.2026
Teemu Virtanen, Projektipäällikkö

Euroopan unionin osarahoittama
Tampereen yliopisto
SEAMK Seinäjoen ammattikorkeakoulu
Vaasan yliopisto UNIVERSITY OF VAASA
ETELÄ-POHJANMAAN LIITTO Regional Council of South Ostrobothnia

Hankkeen ja pilotin taustatiedot



Pilotin tiedot

Seinäjoen Ammattikorkeakoulu ja vAI:lla tuottavuutta? -hanke toteuttivat tuottavuuspilotin yhdessä Plastec Finlandin kanssa.

Pilotin tavoitteena oli evaluoida tekoälyn sopivuutta tuotevirheiden tunnistuksessa ilman perinteistä konenäköteknologiaa ja virheellisten sekä korrektien tuotteiden opettamista erikseen tekoälylle/koneoppimiselle.

Pilotissa tuotetun prototyypin avulla pilotoitiin visuaalisen laaduntarkastuksen automatisointia tuotantoympäristössä. Prototyypin tehtävänä oli tunnistaa, milloin tuote on tarkastusasemassa, ottaa kuvat ja päättää tekoälyn avulla onko tuote hyväksytty vai hylätty.

Käytetyt teknologiat

- Raspberry Pi kappaleiden kuvausta varten
- USB-webkamera (2K-laatu)
- Mistral AI, pixtral-large-latest (huhti-toukokuu 2026) https://docs.mistral.ai/models/model-cards/pixtral-large-24-11
- Myös testattu paikallisesti ajettuina kielimalleina: moondream, nanollava, ministral

- Pilotin yhteydessä luotu ja käytetty Python-ohjelmakoodi on löydettävissä ja käytettävissä avoimesti SEAMKin GitHubissa: https://github.com/SeAMKedu

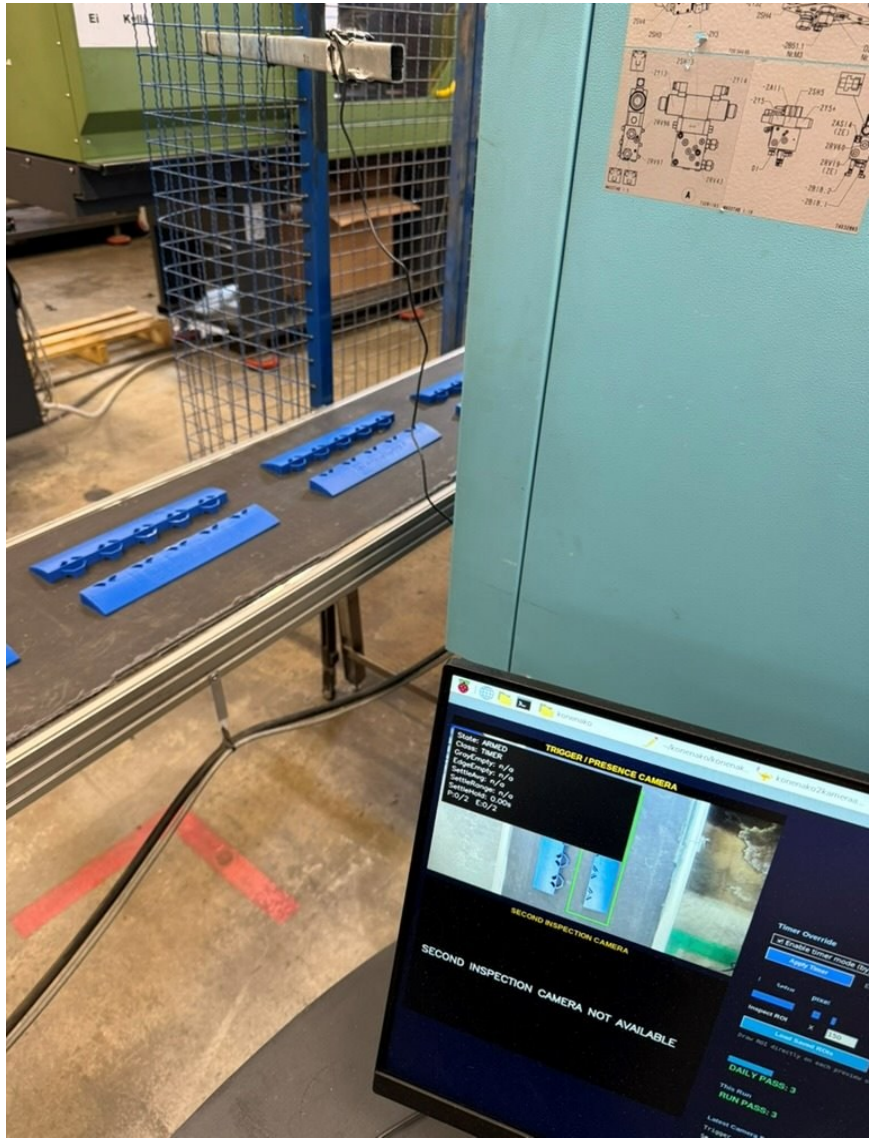
Pilotti käytännössä



Toteutus

Pilotissa toteutettiin seuraava prosessi:

1. **Kuvan ottaminen kappaleesta/tuotteesta**
2. **Kuvan lähettäminen tekoälylle analysoitavaksi**
3. **Hyvien ja huonojen kappaleiden tunnistaminen ja ajankohtaisen tilanteen esittäminen visuaalisessa käyttöliittymässä**



Toteutus

- Python-toteutuksella pyrittiin tunnistamaan tuotteen saapuminen kameranäkymään ja kuvaamaan tuote
  - Kuva lähetettiin joko paikalliselle kielimallille tai API-rajapinnan kautta kolmannelle osapuolelle (Mistral)
- Koska tuotteet liikkuivat nopeasti eivätkä juuri pysähtyneet paikalleen, pilotin prototyyppiin lisättiin ajastin jonka avulla kuva saatiin otettua oikea-aikaisesti
  - Tuotteen tarkka kuvaaminen ilman ajastinta todettiin liian epävarmaksi testauksen jälkeen
- Pilotin yhteydessä tutkittiin Raspberry Pi:llä toimivien paikallisten kielimallien pystyvyyttä tuotevirheiden analysointiin

Tulokset



Rasbperry Pi 5 ja paikalliset kielimallit

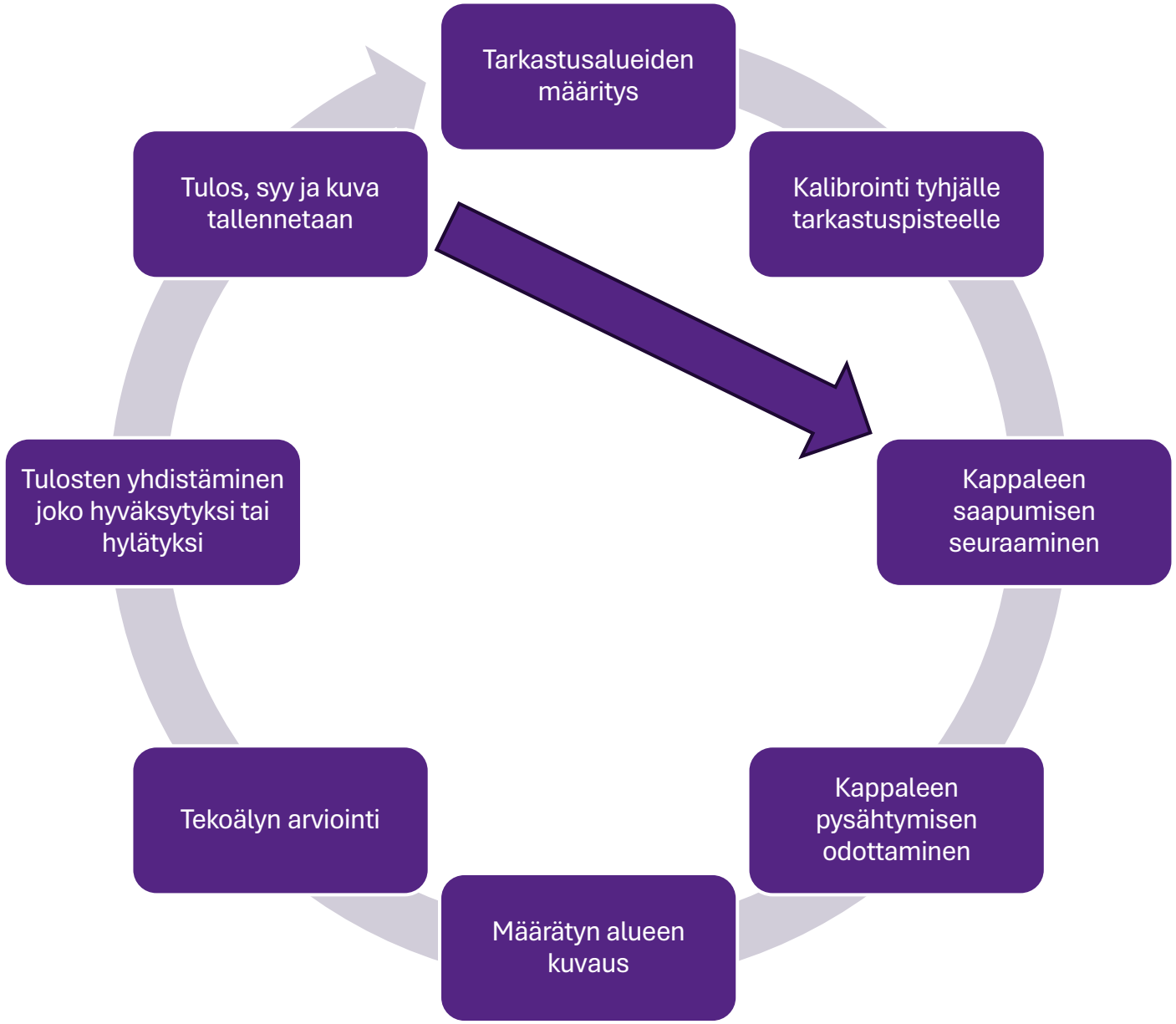
- Kappaleesta piti tunnistaa seuraavat asiat
  - Yhteneväinen väri
  - Ei valuvikoja reunassa (yhteneväinen, pyöreä muoto)
  - Yksi pieni reikä tietyssä kohdassa tuotetta
- Raspberry Pi 5 (8 GB) ja erinäiset pienet ja keskisuuret kielimallit eivät suoriutuneet monivaiheisesta virheentunnistuksesta määritetyssä ajassa (60 sekuntia)
  - Mistral pystyi analysoimaan kuvat ja lähettämään vastaukset muutamissa sekunneissa
  - Paikalliset kielimallit onnistuivaan antamaan vastauksen alle minuutissa, mutta silloin tunnistettavia asioita oli rajattu yhteen tai kahteen



Mistral tunnisti kaksi reikää yhden sijaan ja hylkäsi tämän kappaleen.

Prosessi

- Vain ensimmäisellä käynnistyskerralla vaaditaan tarkistusalueiden määritys ja kalibrointi
- Uudelleenkäynnistyksen yhteydessä nuo tiedot ladataan, eikä alueita tarvitse määritellä uudestaan



Pilotin prototyypistä 1/6

- Pilotin aikana havaittiin, että pelkästään yhden kameran avulla tuotteen reunoja ei voitu tarkastella kokonaisuudessaan, joten prototyyppiin lisättiin mahdollisuus käyttää kahden kameran kuvaa
  - Kahden kameran toiminnallisuus toimii Windows-ympäristössä, Raspberrylle tulee oletettavasti tehdä joitain kevyitä muutoksia
  - Molemmat kuvat lähetetään erikseen tekoälylle (Mistralille), joka on ohjeistettu kehotteessa tarkastelemaan kuvia itsenäisesti toisistaan
- Kuvan ottamisen voi tehdä joko liiketunnistuksen avulla tai ajastimella
- Kuvan tai kuvien analysointiin ja vastauksen saamiseen kuluu muutamia sekunteja ulkoista kielimallia käyttämällä

Pilotin prototyypistä 2/6

- Järjestelmän toimintavaiheet ovat seuraavat:
  - SET ROI: Valitaan tarkastusalue kuvasta
  - CALIBRATE: Tallennetaan tyhjä lähtötilanne
  - ARMED: Järjestelmä odottaa kappaletta
  - TRANSITION: Kuvassa on muutosta, mutta tilanne ei ole vielä varma
  - PRODUCT_MOVING: Tuote havaittu, mutta se liikkuu
  - PRODUCT_STILL: Tuote on paikallaan ja valmis kuvattavaksi
  - ANALYZING: Kuva lähetetään tekoälylle
  - COOLDOWN: Tarkastuksen jälkeen odotetaan hetki ennen uutta tarkastusta
  - WAIT_RESET: Odotetaan, että tarkastusalue palautuu tyhjäksi

Pilotin prototyypistä 3/6

- Koodi tukee tavallisia usb-kameroita
- Järjestelmä yrittää avata kamerat automaattisesti ja mukautuu sen mukaan, onko käytössä yksi vai kaksi kameraa
- Ohjelmassa on graafinen käyttöliittymä, jossa käyttäjä näkee kamerakuvat, järjestelmän tilan, viimeisimmät tarkastustulokset sekä päivän ja ajon tilastot
- Käyttöliittymän kautta määritetään tarkastusalueet, kalibroidaan tyhjä lähtötilanne ja seurataan tuloksia reaaliajassa
  - Hylättyjen kappaleiden kuvat tallennetaan, hyvistä kappaleista voidaan tallentaa joka x:nnes kuva

Pilotin prototyypistä 4/6

- Tekoälyn tarkastusvaiheessa tekoälyn pitää arvioida näkyvä tuotteen osa, etsiä selviä virheitä ja palauttaa yksinkertainen tulos
- Tulos on käytännössä joko **PASS** (tuote hyväksytty) tai **FAIL** (tuote hylätty)
- Lisäksi tekoäly palauttaa lyhyen syykoodin:
  - **OK:** ei virhettä
  - **Missing_pinhole:** Vaadittu reikä puuttuu
  - **Rim_damage:** Reunassa vaurio
  - **Rim_incomplete:** Reuna on puutteellinen
  - **Structural_damage:** Halkeama tai rakennevirhe
  - **Color_not_ok:** Väripoikkeama
  - **Uncertain:** Tekoäly ei pysty arvioimaan kappaletta luotettavasti
  - **Defect_other:** Muu virhe

Pilotin prototyypistä 5/6

- Jos käytössä on kaksi kameraa, molemmat kameranäkymät tarkastetaan erikseen. Lopullinen päätös on PASS eli hyväksytty ainoastaan, jos molempien kameroiden kuva on hyväksytty
- Jokaisesta tarkastuksesta muodostetaan lokirivi
  - Tulokset tallennetaan csv-muodossa, samalla ylläpidetään päivittäisiä kokonaismääriä
- Lokirivien tiedot ovat seuraavat
  - **AIKALEIMA**
  - **ID:** Yksilöllinen tunniste
  - **PASS/FAIL**
  - **VIRHEKOODI JA SELITE**
  - **KAMERAKOHTAISET TULOKSET:** Mikä kamera antoi virheellisen arvion
  - **KUVAN TALLENNUSTIETO**
  - **MITTAUSARVO**

Pilotin prototyypistä 6/6

- Prototyyppi on rakennettu tekoälyavusteisesti käyttäen eri tekoälypalveluita koodauksen yhteydessä, koska pilotin tavoitteena oli evaluoida tekoälyn käytettävyyttä Zero-shot -tyyppisessä laatutarkistuksessa eikä suoraan tuotantokelpoisen sovelluksen rakentaminen
- Prototyypissä otettiin kuitenkin huomioon laatu, jäljitettävyys, operointi, luotettavuus ja hylkäysten auditointi
  - Hylkäyksille on standardoidut syykoodit, jokainen tarkastus kirjataan yksilöllisesti ja aikaleimalla, käyttöliittymä on selkeä ja hylättyjä kuvia voidaan tarkastella jälkikäteen

Pilotin kehotteesta

- Kehotteella pyrittiin ohjaamaan tekoälyä tarkastamaan vain valitun tuotteen näkyvä osa ja palauttamaan vakioitu vastaus: PASS / FAIL +syykoodi
- Tekoälymalli ja taustalla toimiva ohjelmistokoodi voidaan pitää samana, mutta kehotetta muuttamalla järjestelmä voidaan laittamaan tarkastamaan eri tuotteita ja eri laatukriteerejä ilman, että koko sovellusta pitäisi rakentaa uudelleen
- Nykyinen logiikka kertoo tekoälylle seuraavat asiat
  - Mitä tuotetta tarkastetaan
  - Mitä aluetta kuvassa arvioidaan
  - Mitä virheitä etsitään
  - Mitä asioita ei saa tulkita virheeksi
  - Missä muodossa vastaus palautetaan
- Muokattavia asioita tuotekohtaisesti ovat
  - Tuotekuvaus: Mitä kappaletta tunnistetaan
  - Hyväksymiskriteerit: Miltä ehjä tuote näyttää
  - Virhetyypit: Esim. Halkeama, puuttuva osa, muotovirhe
  - Kameranäkymä: Tarkastetaanko kokonaista tuotetta vai vain osaa siitä
  - Syykoodit: Itsemääritellyt laatuluokittelut

Pilotissa käytetty kehote

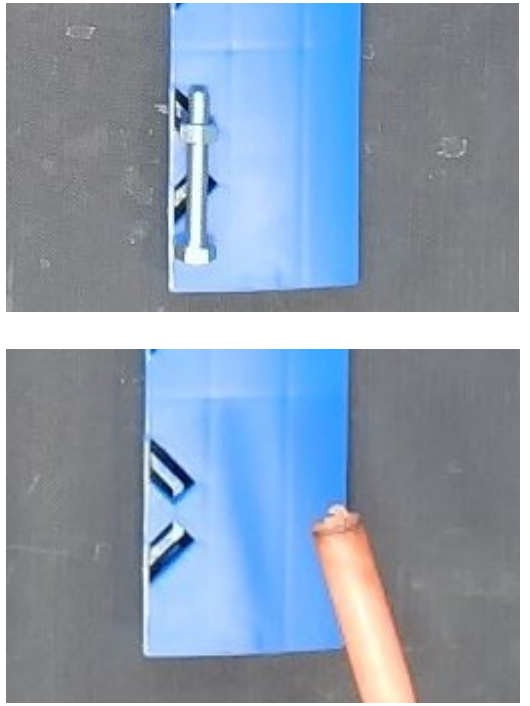
- Inspect one camera view for the target rectangle-shaped product. This is the {view_label} view only.
  First determine whether the target product is actually present in this image.
  If the target product is not present, return FAIL with reason code od.
  Ignore background.
  Return FAIL only if the visible part shows a real defect, such as chip, crack, deformation, incomplete moulding, missing material, structural damage, or if the target product is absent. If the visible segment is too blurry, blocked, cropped incorrectly, or cannot be judged reliably, return FAIL with reason code un.

Tulosten alustus

- Pilotin prototyyppiä rakennettiin keskittyen yksinkertaiseen, ympyränmuotoisen kiekon evaluointiin
- Tässä käyttötapauksessa kehote oli suhteellisen helppo määritellä: tekoälyn piti arvioida näkyvän reunan eheyttä, muotovirheitä ja väriä
- Pilottitestauksessa kuitenkin testattiin prototyyppiä monimutkaisemman tuotteen kanssa
- Kaksoiskamerajärjestelmä tunnisti yksinkertaisissa tuotteissa valuviat ja vierasesineet lähes poikkeuksetta
- Monimutkaisten tuotteiden ominaisuuksien sanoittaminen osoittautui haasteelliseksi
  - Selkeät poikkeamat, kuten vierasesineet ja vääränväriset ainespoikkeamat tekoäly kuitenkin tunnisti

Tulosten läpikäynti 1/3

- Yksinkertaisissa kappaleissa, joissa muoto on yhteneväinen ja rakenneviat selkeitä, viat voidaan tunnistaa lähes aina
- Monimutkaisten kappaleiden osalta pystyttiin tunnistamaan suuria poikkeamia, kuten ylimääräisiä esineitä hihnalla pilotin aikana poikkeuksetta
  - Pilotin aikana ei pystytty testaamaan valmistusvikaa tässä tietyssä kappalessa



Ylimääräisiä esineitä tai objekteja kuvatussa tuotteessa, jotka tekoäly tunnisti tuotteeseen kuulumattomiksi esineiksi.

Tulosten läpikäynti 2/3

- Tekoälyn väärät havainnot ovat lähes poikkeuksetta virheiden havaitsemisen epäonnistumisia, eivätkä virheiden hallusinointeja
- Haastavin kohta tätä koko prosessia on kuvailla tutoe tekoälylle sanallisesti, sekä kuvailla tarkistettavia kohtia sanallisesti
  - Tämä ei ole mahdotonta, mutta mitä monimutkaisemmaksi kuvailu muodostuu, sitä herkemmin tekoäly tekee virheitä
  - Tiettyihin asioihin, kuten esimerkiksi värin tunnistamiseen on varmasti parempia vaihtoehtoja kuin tekoälyn käyttö
  - Poikkeamien tunnistamista varten on myös olemassa muita tapoja, pilotissa haluttiin pilotoida tekoälyn käyttöä ilman koulutusta tai vertailumateriaalia

Tulosten läpikäynti 3/3

- Kokonaisuudessaan voidaan todeta, että tekoäly pystyy tunnistamaan poikkeavuuksia kuvatuissa kappaleissa tietyin rajoittein, vaikka tekoälylle ei oltaisi annettu koulutusmateriaalia ennakkoon tarkistettavasta kappaleesta
- Pilotin yhteydessä tarkasteltiin useita satoja kuvia hyödyntäen Mistralin pixtral-large –kielimallia
- Kokonaisuudessaan token-kustannus oli joitain euroja pelkästään kuvien tunnistamista varten
  - Kehotteen sekä lopullisen vastauksen tokenien määrä verrattuna kuvien kuluttamaan tokenmäärään on hyvin pieni

Prototyypin hyötyjä ja rajoitteita

- Hyödyt
  - Tunnistaa selkeät virheet tehokkaasti
  - Kaksoiskamera parantaa havaintovarmuutta
  - Tilastoitavat tuotantomäärät ja virhemäärät
  - Ei tarvitse kytkentää osaksi muita järjestelmiä vaan toimii itsenäisesti
  - Sama järjestelmä voidaan mukauttaa eri tuotteille kehotetta muuttamalla
- Rajoitteet
  - Monimutkainen tuote tai kappale on vaikea kuvailla tekoälylle sanallisesti
  - Puuttuvien yksityioskohtien tunnistus vaatii tarkat kriteerit
  - Valaistus ja kuvakulma vaikuttavat lopputulokseen
  - Kehotetta tulee optimoida iteroiden
  - Epävarmoissa tapauksissa järjestelmä hylkää kappaleen varmuuden vuoksi

Jatkokehitysideat



Jatkokehitysideat

- Tuotekohtaiset kehotepohjat
  - Eri tuotteille omat tarkastusohjeet, virhetyypit ja syykoodit
- Referenssikuvat
  - Voidaan hyvin vahvasti olettaa, että referenssikuvilla saadaan parempia tuloksia
- Kriittisten tarkastuskohtien itsenäinen tarkistusosio tekoälyllä
  - Oma vaiheensa yhden tietyn kohdan tarkistamiselle
- Parempi kamerakalibrointi
- Integraatio tuotannon järjestelmiin



Euroopan unionin osarahoittama

SEAMK
Seinäjoen ammattikorkeakoulu
Seinäjoki University of Applied Sciences

Tampereen yliopisto

Vaasan yliopisto
UNIVERSITY OF VAASA

ETELÄ-POHJANMAAN LIITTO
Regional Council of South Ostrobothnia