Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

MAIJA SJÖVALL

Tekoälyn hyödyntäminen tarjous-laskennassa

SÄHKÖ JA AUTOMAATIOTEKNIIKAN TUTKINTO-OHJELMA
2024

TIIVISTELMÄ

Sjövall, Maija: Tekoälyn hyödyntäminen tarjouslaskennassa
Opinnäytetyö, AMK
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Toukokuu 2025
Sivumäärä: 45

Opinnäytetyön tarkoituksena oli tutkia tekoälyn hyödyntämistä sähköurakoitsijan tarjouslaskennan tukena. Työssä kehitettiin kuvantunnistusohjelma, Python-ohjelmointikielellä, käyttäen OpenCV-kirjastoa sekä Teplate Matching-menetelmää. Ohjelma tulkitsi pistepiirustusta, josta se pyrki etsimään tiettyjä symboleita kuvantunnistuksen avulla. Tavoitteena oli tunnistaa symbolit pistepiirustuksesta, laskea symbolien lukumäärät sekä merkitä löydetyt symbolit tulostettavaan kuvaan käyttäjää varten.

Template Matching-menetelmä soveltui parhaiten selkeästi erottuvien symbolien kuten kattovalaisimien tunnistamiseen. Tarkkuutta heikensivät yksinkertaiset ja toisiaan muistuttavat symbolit, kuten erilaisten pistorasioiden symbolit. Tunnistuksen tarkkuutta saatiin parannettua ottamalla huomioon erilaiset symbolien skaalaus- ja rotaatiovaihtoehdot. Silti menetelmässä esiintyi haasteita, jotka edellyttivät manuaalista tarkastusta ja korjausta.

Johtopäätöksenä todettiin, että vaikka Template Matching-menetelmä osoitti potentiaalia tarjouslaskennan tueksi, se ei yksin riitä luotettavaksi komponenttien laskentamenetelmäksi ja laskennan automatisoimiseksi. Menetelmän suurimmat rajoitukset liittyivät symbolien samankaltaisuuteen ja pienten erojen erottelukykyyn. Jatkossa kuvantunnistuksen tarkkuutta voisi mahdollisesti parantaa kehittyneemmillä tekoälymenetelmillä kuten syväoppivien neuroverkkojen avulla.

ABSTRACT

Sjövall, Maija: Utilizing artificial intelligence in proposal preparation
Bachelor's thesis
Degree programme in electrical and automation engineering
May 2025
Number of pages: 45

The purpose of this thesis was to investigate the utilization of artificial intelligence to support cost estimation in the electrical contracting field. In this thesis an image recognition program was developed using the Python programming language, the OpenCV library, and the Template Matching method. The program interpreted electrical drawing, from which it aimed to detect specific symbols with the help of image recognition. The goal was to identify the symbols in the electrical drawing, count their occurrences, and mark the detected symbols on the image for the user.

The Template Matching method was found to be best applied to identify clearly distinguishable symbols, such as ceiling lamps. The accuracy was decreased by simple and visually similar symbols, such as different types of socket symbols. The recognition accuracy was able to be improved by considering different variations of scaling and rotation. However, the method still faced challenges that required manual inspection and correction.

The conclusion of this thesis was that although the Template Matching method showed potential as a support tool for proposal preparation, it is not in itself a sufficient tool for reliably counting components or automating the estimation process. The main limitations of the method were related to the similarity of symbols and its limited ability to distinguish small differences. In the future, the accuracy of image recognition could be improved by using more advanced artificial intelligence methods, such as deep learning neural networks.

SISÄLLYS

1 JOHDANTO

Sähköurakoinnissa tarjouslaskenta on kriittinen osa yrityksen toimintaa, sillä se vaikuttaa suoraan yrityksen kilpailukykyyn ja kannattavuuteen. Tarjouslaskennan tarkkuus ja tehokkuus ovat olennaisia, sillä virheellinen tai puutteellinen laskenta voi johtaa joko tarjouskilpailun häviämiseen tai taloudellisesti kannattamattoman urakan vastaanottamiseen. Tarvittavien tarvike- ja työmäärien selvittäminen vie paljon aikaa ja siihen sisältyy paljon rutiininomaista laskentaa.

Jos tarjouslaskentaa saataisiin automatisoitua tekoälyn avulla, se parantaisi huomattavasti yrityksen kilpailukykyä. Automatisointi toisi yritykselle nopeampaa ja tarkempaa tarjouslaskentaa, joka voisi johtaa suurempaan menestykseen tarjouskilpailussa. Automatisointi vapauttaisi yrityksen työntekijöitä keskittymään vaativampiin tehtäviin eli se parantaisi resurssien käyttöä ja työn tuottavuutta.

Kun laskentaa suoritetaan manuaalisesti, voi laskennassa esiintyä inhimillisiä virheitä. Nämä virheet voivat aiheuttaa merkittäviä taloudellisia tappioita yli- tai alihintaisen tarjouksen muodossa. Tekoäly voisi vähentää näitä riskejä. Jos tekoälyavusteinen tarjouslaskenta saadaan luotettavaksi ja tarkaksi, saadaan sen avulla tarjouksista taloudellisesti kannattavia, laskentaan käytettävä aika pienenee ja työvoimakustannukset vähenevät.

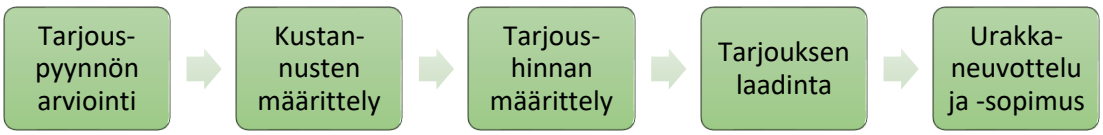
Tässä opinnäytetyössä tutkitaan, miten tekoälyyn perustuvaa kuvantunnistusta voidaan hyödyntää sähköurakoitsijan tarjouslaskennassa. Opinnäytetyön tavoitteena oli kehittää käytännönläheinen kuvantunnistusohjelma, joka kykenee tunnistamaan erilaisia sähkökomponentteja piirikaavioista, laskemaan tunnistettujen komponenttien määrät sekä esittämään tulokset selkeästi käyttäjälle. Työ rajattiin koskemaan yhtä pistepiirustusta, josta etsittiin erilaisia

komponentteja. Kaapelit ja johdotukset rajattiin työn ulkopuolelle, koska ne ovat kaavioissa erisuuntaisia ja -mittaisia, mikä vaikeuttaa luotettavaa kuvantunnistusta.

2 TARJOUSLASKENTA

Sähköalalla kilpailu on kovaa ja vain pieni osa tarjouksista johtaa tilaukseen. Tästä syystä tarjousprosessi pitää olla sujuvaa ja laadullisesti riittävän korkeatasoista. Hinnat on aseteltava kilpailukykyisiksi ja tarjouslaskennan on oltava tarkkaa. Jos näin ei ole, virheellinen hinnoittelu voi johtaa joko tarjouskilpailun häviämiseen tai kannattamattoman työn vastaanottamiseen. (Saastamoinen & Autio, 2017, s. 16–17-)

Tarjouslaskenta voidaan nähdä prosessina, joka etenee tietyssä järjestyksessä vaihe vaiheelta, kuten kuviossa 1 esitetään. Sähköurakoitsijan on hallittava oman alansa tekniikka ja pystyttävä suunnittelemaan asiakkaan tarpeisiin sopiva ratkaisu. Tarjouspyynnön onnistumisen kannalta pitää saada ensin kattava käsitys hankkeesta ja sen yksityiskohdista. Lopputuloksena on hankkeen kokonaishinta. (Saastamoinen & Autio, 2017, s. 17.)



Kuvio 1. Tarjouksen laadinta (Saastamoinen & Autio, 2017)

2.1 Tarjouspyynnön arviointi

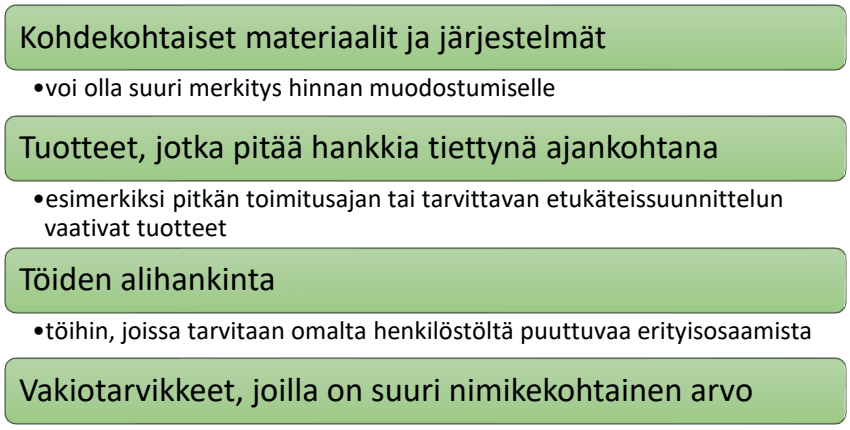
Tarjouslaskenta aloitetaan aina tarjouspyynnön huolellisella arvioinnilla. Ensimmäiseksi on tärkeää selvittää, kuinka hyvin tarjouspyynnön kohde sopii yrityksen tuotantoon ja resursseihin. Arvioinnissa on otettava huomioon kolme keskeistä tekijää. Ensimmäisenä tulee tarkastella henkilöresursseja eli kartoitetaan, onko yrityksellä riittävästi työntekijöitä käytettävissä kyseisenä ajankohtana. Toiseksi on arvioitava, löytyykö yrityksestä tarvittava osaaminen hankkeen toteuttamiseksi. Kolmanneksi on syytä pohtia, tarjoaako toimitus yritykselle jonkin erityisen kilpailuedun. Kun nämä asiat on selvitetty, päätetään urakkamuoto sekä sopimusehdot. Ennen kuin laskentatyö aloitetaan, määritellään tarjouspostit eli kokonaisuudet, joihin työ- ja tarvikemäärät kootaan. Tämä tehdään yleensä sen perusteella, miten osahinnat on pyydetty tarjouspyynnössä. (Saastamoinen & Autio, 2017, s. 18–20.)

2.1.1 Sähköjärjestelmänimikkeet

Työ- ja tarvikemäärät sekä massat on hyvä koota mahdollisuuksien mukaan S2010-nimikkeistön mukaisiin kokonaisuuksiin. Talonrakennusalan käyttöön suunniteltu S2010-sähkönimikkeistö perustuu järjestelmäjaotteluun, jossa hanke jaetaan selkeisiin järjestelmänimikkeisiin. Sähkönimikkeistön hierarkkinen rakenne varmistaa, että ylemmät tasot kattavat aina alemmat kokonaisuudet täysin (kuvio 2). Tällä menetelmällä voidaan kerätä tunnuslukuja, jotka helpottavat uusien, samankaltaisten kohteiden arviointia. (Saastamoinen & Autio, 2017, s. 22.)

Kuvio 2. Sähkönimikkeistön jaottelu (Saastamoinen & Autio, 2017)

Tarjouspyynnön arviointivaiheen lopuksi määritellään ne hankinnat, joista aiotaan pyytää tarjouksia suoraan tuotteen tai palvelun tarjoajilta. Nämä voivat olla kokonaisuuksia, jotka suunnitellaan hankittavaksi valmiina, tai kohteita, joissa on tavanomaisesta poikkeavia erityisvaatimuksia. Valintakriteerit voivat olla kuvion 3 mukaisia. (Saastamoinen & Autio, 2017, s. 22.)



Kuvio 3. Arvioinnin erityisvaatimukset (Saastamoinen & Autio, 2017)

2.2 Tarjoushinnan muodostuminen

Tarjouslaskennan työläin vaihe on tarvittavien tarvikkeiden ja työmäärien selvittäminen. Massalaskennan virheet voivat kertautua ja olla toteutusvaiheessa vaikeasti korjattavissa. Kertolaskuun perustuva arviointi, kuten vertailu euroina per neliö tai asunto, sisältää suuren virheriskin. Asiakirjoista järjestelmittäin mitattava yhteenlaskuperusteinen massalaskenta on huomattavasti tarkempi ja luotettavampi laskentamenetelmä. (Saastamoinen & Autio, 2017, s. 23.)

2.2.1 Tarvikkeiden hinnat ja määrät

Urakkatarjouksen välittömät kustannukset määritellään ensin mittaamalla ja laskemalla tarvikkeet, joita urakassa tarvitaan. Piirustuksista voidaan mitata esimerkiksi kaapeleiden ja johtoteiden pituudet, kun taas laskettavia suureita ovat asennettavien tarvikkeiden määrät, kuten kytkimet ja pistorasiat.

Laskemisessa tulee huomioida myös hävikki ja työvarat. (Saastamoinen & Autio, 2017, s. 28.)
Kappale- tai metrihintaa kutsutaan sähköalan urakkatöissä yksikköhinnaksi. Hinnoittelu voidaan tehdä yrityksen omalla tuotehinnastoluettelolla tai Sähköurakan yksikkökustannukset – kirjan mukaan. Tätä kirjaa ylläpitää Sähköinfo. (Saastamoinen & Autio, 2017, s. 44.) Kirjan mukaan tarvikekustannuksiin on laskettu 7,5 %:n hävikki, ja tarvikkeiden hinnat perustuvat suurimpien tukkuliikkeiden sähkötarvikehinnastoihin (Sähköinfo oy, 2024).

2.2.2 Tarjouslaskentaohjelmat

Markkinoilla on saatavilla erilaisia CAD-ohjelmistoja, joita voidaan käyttää joiltain osin massalaskentaan. Jos ohjelmistoja hyödynnetään oikein, nämä ohjelmat nopeuttavat laskentaa ja vähentävät inhimillisten virheiden riskiä. Laskentasovelluksella laaditaan rakennuskohteelle positiokohtainen tarjoushinta tarjouspyynnön mukaisesti. Positioinnin ohella voidaan käyttää hankeosia tai rakennustiloja. Tarjoushinta muodostetaan positioittain kumuloituvista kustannuslajikohtaisista nettohinnoista, joihin lisätään lakisääteiset ja muut lisäkustannukset. (Saastamoinen & Autio, 2017, s. 23–24.)

Massalaskentaa helpotetaan monipuolisilla tuoterekisterin hakumahdollisuuksilla. Toimialakohtaiset syöttösivut mahdollistavat massoittelun ilman tuotekoodien tai hakunimien ulkoa muistamista. Lisäksi voidaan hyödyntää valmiita tuotepaketteja, jotka nopeuttavat laskentaa. Niitä voidaan ylläpitää esimerkiksi tuoterekisterissä tai omissa tietokannoissa. Tuotepaketti voidaan purkaa rakenteiksi ja töiksi. Ohjelmiston hinnoittelutoiminto etsii rakenteille ja töille päivitetyt hinnat suoraan tukkureiden sivuilta tai ohjelmatoimittajien hinnastopäivityspalvelusta. Hinnoittelu tarkistaa nollahintaiset ja -määräiset rivit sekä siirtää ne puutelistalle. Hinnoittelu sisältää myös työmaan erilliskustannuksen laskennan, joka jakaa kulut kullekin positiolle positioiden nettohintojen suhteessa. Tarjous voidaan hinnoitella uudelleen milloin tahansa, ja lopullista tarjoushintaa voidaan tarvittaessa muokata positiokohtaisesti myös käsin. (Saastamoinen & Autio, 2017, s. 23–24.)

2.2.3 Työn hinnoittelu

Kun tarvikkeiden määrä on selvillä, saadaan työ hinnoiteltua. Sähköalan työn hinnoitteluun löytyy valmiita ohjeistuksia. Näitä ohjeistuksia ylläpitävät Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry sekä Sähköalojen ammattiliitto ry. Hinnoittelu perustuu sähköistysalan työehtosopimuksiin, joista löytyy urakkahinnoittelutaulukot. Taulukoista löytyvät eri töiden hinnat, jotka on lajiteltu selkeästi työtehtävien mukaan. (Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry, 2023).

Työehtosopimuksesta löytyy esimerkiksi valaisimien ja niiden erillisten liitäntälaitteiden asennus sekä kytkentä. Asennushinta määräytyy asennus- tai kiinnitystavan, painon sekä pituuden perusteella. (Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry, 2023). Sähköliiton sivulta löytyy myös urakkalaskentaesimerkkejä kuten kuvassa 1. (Sähköliitto, 2024).

Kuva 1. Upotettavien valaisimien asennus (Sähköliitto 2024).

Kuvan 1 valaisimille saadaan asennuksen ja kytkennän hinnat työehtosopimuksen taulukoiden 1 ja 2 mukaisesti. Valaisimen massa on 0,6 kg ja se saadaan kiinnitettyä ilman työkalua. Jos valaisimia asennetaan 5 kappaletta,

asennuksen hinnaksi tulee 5 · 3,43 € eli 17,15 €. Valaisimen kytkentäkaapeli koostuu kolmesta 1,5 mm^2 johtimesta, jolloin yhden valaisimien kytkentä on 0,73 € ja viidelle lampulle hinnaksi tulee 10,95 €. Taulukosta 3 saadaan jakorasian hinnaksi 10,66 €. Eli yhteensä valaisinten työn hinnaksi tulisi 38,76 €.

Taulukko 1. Valaisimien asennus (Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry., 2023)

Rivi	Suurin sivun pituus (cm)	Valaisimen paino enintään (kg)	Kiinnitys työkalulla (€/kpl)	Kiinnitys ilman työkalua (€/kpl)	Kiinnittämättä (€/kpl)
11	175	4	4,62	3,43	2,62
12	175	7	6,23	4,8	3,79
13	175	12	7,12	5,69	4,66

Taulukko 2. Valaisimien kytkentä (Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry., 2023)

Rivi	Toimenpide	Hinta (€/kpl tai €/m)
11	Valaisimen / liitäntälaitteen kytkentä enintään 2,5 mm² / johdin	0,73
12	Valaisimen/liitäntälaitteen kytkentä enintään 4–6 mm² / johdin	1,01
13	Valaisimen/liitäntälaitteen ryhmäjohdon yhdistäminen pistoliittimellä, enintään 2,5 mm²	1,16
14	Pienoispistoliittimen yhdistäminen enintään 1,0 mm²	0,73
15	MCMK-, FRHF- ja vastaavien johtojen pään kytkentäkuntoon valmistaminen	3,6
16	Lisähinta valaisimesta, jossa on myös pistorasia/kytkin	1,56
17	Lumi- tai tippuvesisuojan asennus	1,46
18	Valaisimen yli 4 irrallista osaa, €/kpl	0,53
19	Valaisimen alle asennettavan kiskon asennus ja katkaisu/m	3,98
20	Lisähinta valolähteen asennuksesta vesitiiviisiin pitimiin	0,9
21	Valaisinkohtainen johtimien sukitus	1,19
22	Valaisimen ripustustangon, ketjun tai vaijerin katkaisu ja valmistaminen	0,8
23	Valaisimen suojamuovin/kalvon poistaminen / valaisin	0,8

Taulukko 3. Jakorasioiden asennus ja kytkentä (Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry., 2023)

Rivi	Johtimen poikkipinta-ala (mm^2)	Uppo/ puu (€/kpl)	Kivi, metalli (€/kpl)	Haaroitusrasia kytkentäpistoliittimillä (€/kpl)
12	1,0–2,5	10,66	12,11	6,9
13	6	11,55	12,99	

2.2.4 Muut kustannukset

Urakan yhteydessä tehdään usein myös sellaisia töitä, joita ei voida hinnoitella suoraan urakkahinnoittelun mukaisesti, vaan niiden kustannukset on arvioitava erikseen. Yleensä erilliskustannukset ovat noin 20–25 % koko urakan muusta arvosta. Näiden kustannusten selvittäminen ja arviointi on olennainen osa työkohteen tarvittavien töiden ja tarvikkeiden määrittelemisessä. Taulukkoon 4 on listattu erilliskustannuksia ja miten ne voidaan ottaa huomioon tarjousta laatiessa. (Saastamoinen & Autio, 2017, s. 34–40).

Taulukko 4. Erilliskustannusten huomioiminen tarjouslaskennassa (Saastamoinen & Autio, 2017)

Erilliskustannus	Huomioitavat asiat laskennassa
Purkutyöt	Purettavan materiaalin poiskuljetus, kaatopaikka- ja ongelmajätemaksut.
Hankalat olosuhteet	Kohteen sijainnista riippuvat erityisolosuhteet.
Aputyöt	Esimerkiksi telineiden rakennus tai sähköasennuksiin liittyvien tarvikkeiden siirtäminen, jos ne eivät kuulu pääurakkaan.
Ylityöt	Normaalin työajan ulkopuoliset työt joko sisällytettävä tai rajattava selkeästi.
Koekäytöt ja käytön opastus	Käytönopastuksen ja koekäyttöjen vaatiman ajan laskenta tarjoukseen.
Työmaatilat	Konttori-, kalusto- ja muut tilat, hinnoittelu vuokrahintojen mukaan.
Kuljetukset ja varastointi	Kuljetus-, nosto- ja varastointikulut arvioitava vuositasolla.
Työkalut ja telineet	Telineiden ja erikoistyökalujen vuokra- ja lainakustannukset.
Vakuutukset	Urakkaan sisältyvät vakuutukset kuten palo- ja varkausvakuutukset.
Suunnittelu	Työ- ja käyttöpiirustusten sekä dokumentoinnin kustannukset (2–3 %).
Tarkastukset	Esimerkiksi viranomais-, vastaanotto- ja varmennustarkastukset.
Jälkityöt	Jälkitöiden kustannusten varaus (esim. puutelistan mukaan).
Takuutyöt	Keskimääräinen takuutyökustannus noin 0,1–0,5 %.
Vakuutusprovisiot	Urakan vakuudet ja ennakkomaksujen palautustakaukset.
Työmaan valmistelu ja tilapäiset asennukset	Valmistelut ja tilapäiset sähköasennukset.
Riskit ja rahoituskulut	Mahdollisten riskivarausten huomioonottaminen.
Kärkimieslisä	Työmaan kärkimiehen TES:n mukainen korvaus.
Työkohteen sijainnista johtuvat erilliskustannukset	Ateriakorvaukset, päivärahat, matkakustannukset, matka-ajan palkat, majoitus, yöpymiskorvaus ja juhlapyhien kotimatkat.
Työnjohtokustannukset	Työnjohdon palkat, ruokailu-, majoitus- ja matkakulut.
Sosiaalikustannukset	Työlainsäädännön ja TES:n mukaiset kustannukset.

2.3 Tarjouksen laadinta

Tarjouksen laatimisessa tulee ottaa huomioon katetarve sekä määrittää kokonaishinta. Keskimääräinen myyntikate määritellään vuosibudjetissa, mutta

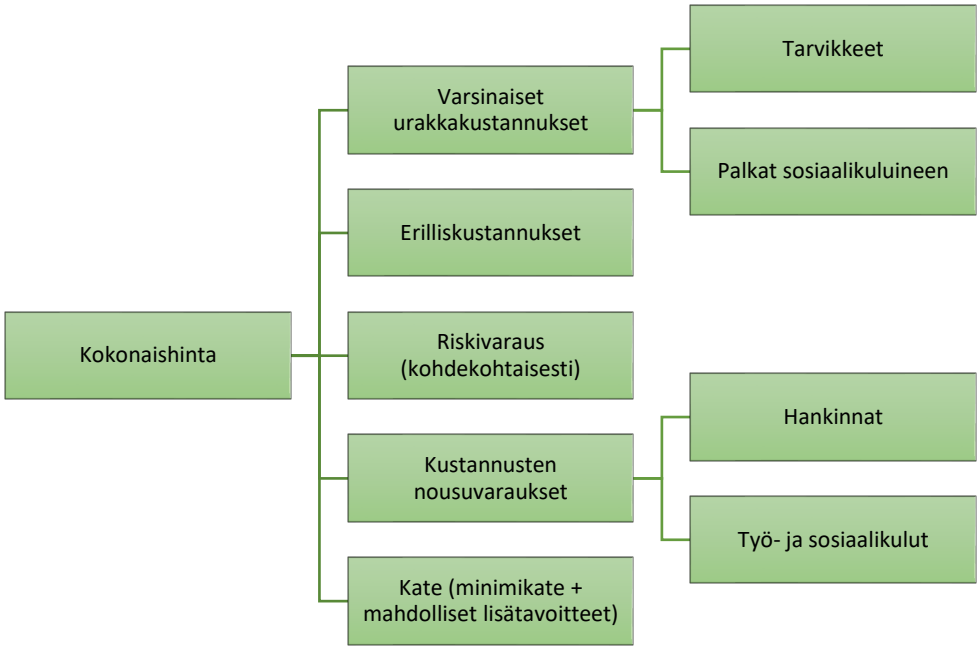
käytännössä kateprosentti voi vaihdella markkinoiden mukaan. Jokainen tarjous on arvioitava erikseen. Tuotteen hinta on oikea silloin, kun sekä ostaja että myyjä kokevat tehneensä hyvän kaupan. Sopiva hinta on yritykselle kannattavan liiketoiminnan perusta, sillä myyntikatteella katetaan kiinteät kulut, pääomakustannukset ja tuotetaan voittoa. (Saastamoinen & Autio, 2017, s. 10, 41.)

Liiketaloudellisesti kannattava tulos saavutetaan oikealla hinnoittelulla. Saastamoinen ja Autio (2017, s. 42) tiivistävät oikean hinnan määrittämisen seuraavasti:

- Välittömien kustannusten selvittäminen tarkasti
- Välillisten kustannusten huolellinen arviointi
- Asiakastarpeiden tunteminen
- Markkinamuutosten ennakointi
- Oman osaamisen tunnistaminen
- Tavoiteltavan taloudellisen tuloksen asettaminen
- Huolellisten laskelmien teko
- Tulosteen laadinta

Urakkatarjouksen kokonaishinta muodostuu kuvion 4 mukaisesti. Taulukossa 5 on esimerkki, miten tarjouksen kokonaishinta voidaan määritellä niin, että kate ja kustannusnousuvaraus ovat erilaiset toimituksen erilaisille kustannusosille.

Kuvio 4. Kokonaishinta (Saastamoinen & Autio, 2017)



Taulukko 5. Myyntihintalaskelma (Saastamoinen & Autio, 2017)

	Kustannukset (€)	**Kustannusnousuvaraukset (%)**	**Kustannusnousuvaraukset (€)**	**Kustannukset yhteensä (€)**	**Kate (%)**	**Kate (€)**	**Yhteensä (€)**
Materiaalit	150 000	3,0	4500	154 500	15,0	27 265	181 765
Projekti hankinnat	125 000		0	125 000	15,0	22 059	147 059
Palkat	140 000		0	140 000	15,0	24 706	164 706
Matka- ja päivärahat	18 000		0	18 000	15,0	3 176	21 176
Alihankinnat	15 000		0	15 000	15,0	2 647	17 647
Yhteensä	**448 000**		**4500**	**452 500**	**15,0**	**79 853**	**532 353**

2.4 Tarjouksesta urakkaneuvotteluun ja -sopimukseen

Tarjous tehdään yleensä tarjouspyynnön pohjalta. Tarjouspyyntö ei vielä sido tilaajaa. Tarjouksen tulee vastata tarjouspyynnön ehtoja, mutta mikäli niistä poiketaan, on erot mainittava selkeästi. Muuten tarjous voidaan hylätä. Rakennusalalla käytetään yleisesti YSE 1998 -vakioehtoja, jotka varmistavat

molempien osapuolten tasapuolisen kohtelun sekä sopimuksen aikana, että mahdollisissa erimielisyystilanteissa. (Saastamoinen & Autio, 2017, s. 47–48.)

Tarjous on vastattava asiakkaan odotuksiin ja tarpeisiin. Tarjouksessa tulee olla tarjoushinta, arvonlisävero, kenelle tarjous on osoitettu, mahdollinen sopimuskumppani, tarjouksen kohde ja mihin se perustuu, mahdolliset poikkeamat, sopimusehdot, tarjouksen voimassaoloaika, maksuehdot sekä yhteyshenkilö. Urakkatarjous sitoo tekijäänsä siitä hetkestä, kun vastaanottaja saa sen tietoonsa. Tarjouksessa on aina mainittava sen voimassaoloaika. Tarjouksen sitovuus päättyy, jos tarjous hylätään, jos tilaaja hyväksyy kilpailevan tarjouksen tai jos tarjouksen voimassaoloaika päättyy ilman hyväksyntää. (Saastamoinen & Autio, 2017, s. 51–52.)

Vaikka urakkaneuvotteluun päästään, se ei vielä tarkoita sitä, että lopullinen sopimus syntyisi. Tilaaja voi esittää tarkentavia kysymyksiä, joihin urakoitsijan on pystyttävä vastaamaan heti. Siksi tarjoavan yrityksen tulee harkita tarkkaan, ketkä yrityksen edustajat osallistuvat neuvotteluun. (Saastamoinen & Autio, 2017, s. 53.)

Urakkaneuvottelujen tavoitteena on selventää tarjouksen sisältöä, varmistaa molemminpuolinen ymmärrys urakan laajuudesta, käsitellä mahdolliset rakennuttajan muutostoiveet sekä sopia yhteisistä toimintatavoista. Neuvotteluista kannattaa laatia huolellinen pöytäkirja, johon osapuolet voivat tarvittaessa palata myöhemmin. Selkeä neuvottelupöytäkirja helpottaa epäselvyyksien ratkaisemista sopimuskauden aikana ja varmistaa, että kaikki tarjoajat saavat tasavertaisen kohtelun. (Taivainen, 2023.)

Kun tilaaja ilmoittaa hyväksyneensä urakkatarjouksen, seuraava vaihe on sopimuksen allekirjoitus. Tässä vaiheessa on tärkeää varmistaa, että urakkaneuvotteluissa sovitut muutokset ja täsmennykset on kirjattu sopimusasiakirjoihin sovitulla tavalla. Allekirjoitusten jälkeen alkavat sovitut työt. (Saastamoinen & Autio, 2017, s. 54.)

3 TEKOÄLY

3.1 Tekoälyn historia ja kehitysaallot

Tyypillisesti koneet ovat suorittaneet raskaita ja toistuvia tehtäviä. 1950-luvulla heräsi ajatus siitä, voisiko ihmisen älykkyyttä simuloida koneiden avulla. Tämä oivallus aloitti tekoälyn kehittämisen. Aluksi keskityttiin asiantuntijajärjestelmiä, jotka tallensivat tietoa tietyltä alalta ja käyttivät asiantuntijoiden luomia sääntöjä. Tiedon keruu ja päivittäminen oli kuitenkin kallista ja hidasta, mikä rajoitti tekoälyn yleistymistä. (Aaltonen, 2019, s. 55.)

Alan Turing julkaisi vuonna 1950 artikkelin Computer machinery and intelligence. Artikkelissaan hän esitteli testin, jonka tarkoituksena oli selvittää, voiko tietokone olla älykäs ja ajatella kuten ihminen. Testiä alettiin kutsumaan Turingin testiksi. (Siukonen & Neittaanmäki, 2019, s. 41.) Testissä ihminen esittää kysymyksiä kirjallisesti ja pyrkii selvittämään, onko vastaaja ihminen vai kone. Jos kysyjä ei pysty erottamaan, tulevatko vastaukset ihmiseltä vai koneelta, katsotaan koneen olevan riittävän hyvä ihmiseksi. (Järvinen, 2023, s. 52.)

John McCarthy, Marvin Minsky, Claude Shannon ja Nathan Rochester järjestivät vuonna 1956 kesäseminaarin Darthmouth Collegessa. Tässä seminaarissa termi artificial intelligence eli tekoäly vakiintui. Seminaarin lähtökohta oli, että oppimista ja älykästä toimintaa voidaan kuvata niin tarkasti, että niitä on mahdollista simuloida tietokoneella. Tapahtumassa käsiteltiin muun muassa kieltä käyttäviä tietokoneohjelmia, uusien käsitteiden muodostamista ohjelmistoissa, automaattista ongelmanratkaisua sekä itseään kehittäviä ohjelmia. Nämä aiheet ovat yhä keskeisiä tekoälytutkimuksessa. (Toivanen, 2023, s. 31–32.)

1990-luvulla alettiin hyödyntämään koneoppimista, jonka tavoitteena oli tunnistaa tilastollisia kuvioita suuresta datamäärästä. Samaan aikaan kehitettiin luonnollisen kielen ymmärtämistä, ongelmanratkaisua, navigointia ja havaintoteknologioita. 2000-luvulla koneoppimista on kehitetty niin, että tilastollisia ja

todennäköisyyspohjaisia menetelmiä voidaan soveltaa entistä suurempiin datajoukkoihin. Tekoälytutkimuksessa on keskitytty syväoppimiseen ja neuroverkkoihin, jotka pystyvät suorittamaan erilaisia luokittelu- ja ennakointitehtäviä hyödyntämällä historiallista dataa. (Aaltonen, 2019, s. 55.)

3.2 Heikko ja vahva tekoäly

Tekoäly voidaan jakaa kahteen luokkaan: heikkoon ja vahvaan. Heikko tekoäly perustuu koneoppimiseen, jossa tietokoneohjelmistot suoriutuvat annetuista tehtävistä algoritmien avulla. (Siukonen & Neittaanmäki, 2019, s. 45.) Se suorittaa tehtäviä ohjelmoidun logiikan mukaisesti, mutta ei pysty laajentamaan osaamistaan muihin tehtäviin (Merilehto, 2018, s. 23). Vahvan tekoälyn tavoitteena on kehittää kone, joka oppisi matkimaan ihmisen aivotoimintaa ja ylittämään ihmisen älylliset kyvyt. Toisin kuin heikko tekoäly, joka keskittyy yksittäisiin tehtäviin, vahva tekoäly pyrkii luomaan järjestelmän, joka toimii itsenäisesti ja ymmärtää oman tilansa. Saavuttaakseen tämän koneen tulisi olla tietoinen itsestään, ymmärtää ympäröivää maailmaa ja asettaa omia tavoitteitaan. Jotta tämä olisi mahdollista, tekoälyn täytyy käsitellä tietoa tavalla, joka perustuu matemaattisiin lainalaisuuksiin ja sen omaan logiikkaan. (Siukonen & Neittaanmäki, 2019, s. 45.)

3.3 Koneoppiminen

Koneoppiminen on tekoälyn osa-alue, jossa tietokone oppii käsittelemään ja luokittelemaan dataa ilman, että sen toimintaa on ennalta ohjelmoitu tarkasti. Koneoppiminen perustuu algoritmeihin, jotka kehittyvät vaiheittain käsittelemänsä datan perusteella. Tämä mahdollistaa entistä tarkempien mallien luomisen, joiden avulla voidaan tehdä ennusteita ja tunnistaa kuvioita tiedosta. Mitä enemmän dataa algoritmi saa, sitä tarkemmin se pystyy analysoimaan ja ennustamaan tuloksia. (Merilehto, 2018, s. 27–28.) Ajan myötä kone oppii muokkaamaan omaa toimintaansa, jolloin se kykenee suoriutumaan monimutkaisemmista tehtävistä (Aaltonen, 2019, s. 158).

Koneoppimisessa data jaetaan kahteen osaan: opetusdataan ja testidataan. Suurin osa aineistosta käytetään mallin koulutukseen ja loput varataan sen testaamiseen. Näin voidaan arvioida, kuinka hyvin malli toimii uusilla, sille ennestään tuntemattomilla tiedoilla. (Merilehto, 2018, s. 29.) Datan määrä ja laatu ovat keskeisiä tekijöitä mallin toimivuuden ja päätelmien luotettavuuden kannalta (Vartiainen ym., 2021, s. 107). Aineistoissa voi olla virheitä ne voivat olla vanhentuneita tai ne eivät vastaa todellisia käyttötapauksia (Toivanen, 2023, s. 85–86).

Koneoppimisessa varmuus viittaa siihen, kuinka luotettavasti opetettu malli pystyy luokittelemaan uutta, aiemmin näkemätöntä dataa. Täydellistä tarkkuutta ei yleensä saavuteta, mutta malli voidaan optimoida toimimaan riittävän hyvin vastaamaan käyttötarkoitusta (Vartiainen ym., 2021, s. 107). Hyvän tarkkuuden saavuttaminen vaatii usein suuria määriä dataa, mutta datan määrän kasvattaminen voi johtaa tasapuolisuuden heikkenemiseen. Lisäksi mallien vertailukelpoisuus aiempiin tutkimuksiin voi kannustaa käyttämään vanhoja aineistoja, vaikka saatavilla olisi uudempia ja edustavampia vaihtoehtoja. (Toivanen, 2023, s. 85–86.) Datan suuren määrän lisäksi tulee varmistaa, että ilmiön taustalla on todellisia riippuvuussuhteita ja että data sisältää riittävästi tietoa riippuvuussuhteiden tunnistamiseen. Lisäksi algoritmin tulee olla tehokas, sillä suuren datamäärän käsittely vaatii paljon muistia ja laskentaa. (Alpaydim, 2021, s.41.)

Koneoppiminen on erityisen hyödyllistä, kun ilmiön tarkka ymmärtäminen on haastavaa tai sen mallintaminen vaatisi liikaa työtä. Toisinaan on huomattavasti helpompaa kerätä suuri määrä opetusdataa kuin yrittää määrittää tarkat säännöt jokaiseen tilanteeseen erikseen. (Vartiainen ym., 2021, s. 106.)

3.4 Tekoälyn kouluttaminen

Tekoälyn kouluttamisessa hyödynnetään yleensä kolmea eri oppimismenetelmää: ohjaamatonta oppimista, ohjattua oppimista ja vahvistusoppimista. Näiden menetelmien ero on siinä, kuinka algoritmi hyödyntää saatavilla olevaa

dataa ja palautetta oppiakseen tunnistamaan malleja, tekemään ennusteita tai optimoimaan toimintaansa.

3.4.1 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa koneelle annetaan ainoastaan dataa ilman valmiiksi määriteltyjä oikeita vastauksia. Algoritmin tehtävä on etsiä datasta malleja, organisoida data itsenäisesti ja havaita datan yhtäläisyyksiä ja eroavaisuuksia. Tätä menetelmää voidaan käyttää erityisesti luokittelutehtävissä, joissa pyritään löytämään tietojen välisiä yhdistäviä tekijöitä ja muodostamaan niistä ryhmiä. Ohjaamaton soveltuu myös datan poikkeavuuksien etsintään. Tällaisia poikkeavuuksia voivat olla esimerkiksi petokset tai virheelliset kirjaukset. (Kananen& Puolitaival, 2019, s. 51, 54)

3.4.2 Ohjattu oppiminen

Ohjatussa oppimisessa haluttu lopputulos tiedetään etukäteen ja malli opetetaan yhdistämään syötedata oikeisiin vastauksiin (Merilehto, 2018, s. 28). Malliin tallennetaan kaikki datasta löydettävät ominaisuudet eli attribuutit, joiden avulla se oppii tunnistamaan yhteyksiä syötteiden ja lopputulosten välillä. Kun malli on koulutettu, sille syötetään uutta dataa ilman valmiita vastauksia. Aiemmin opitun perusteella algoritmi pystyy tekemään päätöksiä ja tuottamaan ennusteita. Ohjattua oppimista voidaan hyödyntää esimerkiksi kuvan tunnistamisessa, laadun valvonnassa ja hinnoittelussa. (Kananen & Puolitaival, 2019, s. 48–50)

3.4.3 Vahvistusoppiminen

Vahvistusoppimisessa algoritmi optimoi toimintaansa monimutkaisessa ympäristössä, jossa sille on määritelty kriteerit toivotulle ja ei-toivotulle toiminnalle (Vartiainen ym., 2021, s. 107). Oppimisprosessi perustuu jatkuvaan kokeiluun ja palautteeseen. Kun algoritmi saavuttaa hyvän lopputuloksen, se vahvistaa toimintatapoja, jotka johtivat siihen. Algoritmi tasapainottelee tuttujen,

toimivien ratkaisujen sekä uusien, mahdollisesti parempien vaihtoehtojen kokeilemisen välillä. Oppiminen ei kuitenkaan ole suoraviivaista, sillä tulokset voivat ilmetä viiveellä ja päätökset voivat vaikuttaa tulokseen epäsuorasti. (Toivanen, 2023, s. 81–82.)

3.5 Neuroverkot

Neuroverkot jäljittelevät ihmisaivojen toimintamalleja ja aivojen rakennetta. Ne ovat matemaattisia laskentayksiköitä, jotka pystyvät oppimaan havainnoimalla ja saavuttamaan asetettuja tavoitteita. (Merilehto, 2018, s. 45, 47.) Tekoälyn keinotekoiset neuroverkot koostuvat yksinkertaisista laskennallisista yksiköistä, neuroneista, joita voi olla muutamasta miljardeihin asti. Nämä neuronit muodostavat verkoston, jossa yhden neuronin tuottama tulos toimii syötteenä seuraaville, jotka jatkavat laskentaa omilla menetelmillään. Neuroverkot käsittelevät tietoa kerroksittain: ensimmäiset neuronit ottavat vastaan syötteen, välissä olevat kerrokset jalostavat sitä ja viimeiset neuronit tuottavat lopullisen tuloksen. (Toivanen, 2023, s. 83–84.) Neuroverkot osaavat muun muassa tunnistaa ja nimetä valokuvista eläimiä, rakennuksia, esineitä ja muotoja (Merilehto, 2018, s. 45–46.)

Neuroverkoilla on tekoälyn kannalta kaksi keskeistä etua. Ensinnäkin, vaikka yksittäiset neuronit ovat yksinkertaisia, verkko kokonaisuutena pystyy suorittamaan erittäin monimutkaisia laskutoimituksia. Toiseksi verkon toimintaa voidaan säätää muokkaamalla neuroneita, jolloin lopputulosta voidaan ohjata haluttuun suuntaan ilman, että ohjelman tekijän tarvitsee tietää tarkkaa laskentatapaa. (Toivanen, 2023, s. 83–84.)

Neuroverkko voi luokitella, onko annetussa valokuvassa kissa vai ei. Tällöin syötteenä toimii valokuvan pikselit ja tuloksena arvo väliltä 0–1. Mitä suurempi arvo on, sitä varmemmin kuvassa on kissa. Neuroverkkoa voidaan opettaa antamalla sille kuvia yksi kerrallaan. Jos annetussa kuvassa on kissa ja verkon tulos ei ole yksi, säädetään neuroneja niin, että tulos saadaan lähemmäksi yhtä. Toistojen avulla verkko oppii tunnistamaan kissan yhä paremmin.

Onnistuneen oppimisen seurauksena neuroverkko muodostaa käsityksen siitä, miltä kissa yleensä näyttää kuvassa. (Toivanen, 2023, s. 83–84.)

Fei-Fei Li, Stanfordin yliopiston tekoälylaboratorion johtaja julkaisi vuonna 2009 ImageNet-tietokannan. Tietokanta sisälsi yli kolme miljoonaa jäsenneltyä ja merkittyä valokuvaa. Seuraavana vuonna Li käynnisti kansainvälisen kilpailun laajentaakseen tietokantaa. Tämän tarkoitus oli kehittää algoritmeja, jotka pystyvät indeksoimaan, hakemaan, järjestämään ja merkitsemään multimediatietoja entistä tarkemmin. Li:n tutkimukset osoittivat, että suurten datamäärien hyödyntäminen on avain kuvantunnistukseen ja kaupallisten sovellusten kehittämiseen. Saman johtopäätöksen teki myös neuroverkkojen asiantuntija Geoffery Hinton. Hän saavutti merkittävästi paremman tarkkuuden kuvantunnistuksessa rakentamalla syviä neuroverkkoja, joissa jokainen kerros oppii tunnistamaan erilaisia piirteitä, kuten muotoja ja rakenteita. Tätä lähestymistapaa alettiin kutsumaan syväoppimiseksi. (Siukonen & Neittaanmäki, 2019, s. 77.)

3.6 Kuvantunnistus

Tekoäly on tehokas työkalu kuvien analysointiin ja kohteiden etsimiseen erityisesti silloin, kun kuvissa esiintyy vaihtelevuutta. Vaihtelevuus voi olla muuttuvia taustoja, osittain peittyneitä kohteita tai erilaisia valaistusolosuhteita. Tällöin tekoälyä voidaan opettaa käyttämällä suurta määrää merkittyjä kuvia, joissa etsittävät kohteet ovat ennalta määritelty. Jos taas kuvasta tarvitaan tarkkoja mittauksia tai on löydettävä yksittäinen, täsmällinen piste, ei tekoäly ole paras ratkaisu. Ainakaan vielä. Tulevaisuudessa tekoäly kehittynee niin, että se pystyy tunnistamaan ja luokittelemaan kohteet sekä määrittämään niiden tarkan poimintapisteen entistä luotettavammin. (Ahonen ym., 2023, s. 184.)

3.7 Tekoälyn mahdollisuudet tarjouslaskennassa

Monilla toimialoilla tarjouslaskenta on edelleen pitkälti manuaalinen prosessi, jossa ihminen analysoi saamansa tiedot ja laatii niiden perusteella tarjouksen. Koska tarjouspyynnöt ovat usein rakenteeltaan samanlaisia, voitaisiin niitä käsitellä automaattisesti hyödyntämällä koneoppimista. (Ala-Rakkola, 2020, s.1).

Tekoäly pyrkii suorittamaan ihmisen älyä vaativia toimintoja, kuten oppimista, ajattelua, havaitsemista, luovuutta ja ongelmanratkaisua. Se pystyy suoriutumaan rajatuista tehtävistä ihmistä nopeammin ja tarkemmin sekä pystyy toistamaan niitä rajattomasti. Voimme ohjelmoida tehtäviä ja prosesseja, joita kone suorittaa itsenäisesti ilman ihmisen apua tai ihmisen valvomana ja täydentämän. Tekoälyllä ei ole muuta mieltään ja on puolueeton, mikäli sen harjoitusdata on puolueeton. (Aaltonen, 2019, s. 150; Kananen & Puolitaival, 2019, s.17, 33, 37.)

Tietokoneohjelmat ovat tehokkaita muistamaan, hakemaan täsmällistä tietoa sekä tekemään loogista ja tilastollista päättelyä. Ne pystyvät analysoimaan ja hyödyntämään suuria tietomääriä. Kone toimii ohjelmoitujen mallien mukaisesti toistaen toimintaohjeita muuttuvissakin tilanteissa, kun taas ihminen luottaa tietoonsa ja kokemukseensa, oivaltaa ja ottaa riskejä. (Siukonen & Neittaanmäki, 2019, s. 44.; Toivanen, 2023, s. 18.)

Tekoälyn hyödyntäminen tarjouslaskennassa tuo merkittäviä etuja, kuten tehokkuuden, tarkkuuden ja kilpailukyvyn. Tekoälyä voidaan hyödyntää sekä tarjouslaskennan prosesseissa että hankintaprosesseissa. (Salo, 2024; Admicom Oyj, 2024.)

3.7.1 Tekoäly osana tarjouslaskennan prosessia

Tarjouslaskenta on monivaiheinen prosessi, jossa kerätään ja analysoidaan tietoa sekä laaditaan tarjous. Useimmiten tarjouslaskennan suurin haaste on aika. Tarjousten laatiminen ja vaatimusten analysointi vaativat paljon manuaalista työtä. Tekoäly tuo tähän helpotusta, sillä se pystyy käsittelemään suuria

määriä historiadataa sekä tunnistamaan trendejä, jotka auttavat ennustamaan tulevia kustannuksia ja hintoja aiempaa tarkemmin. Näin tekoäly pystyy vähentämään inhimillisten riskien määrää ja parantaa laskelmien tarkkuutta. (Salo, 2024.) Tekoäly voi analysoida aiempia projekteja ja oppia niiden tietojen pohjalta, mikä mahdollistaa tarkemmat ennusteet tuleville hankkeille. Se pystyy tunnistamaan kaavamaisia budjettiylityksiä, viivästyksiä tai muita riskitekijöitä, jotka toistuvat tietyissä rakennusvaiheissa tai materiaalien käytössä. (Admicom Oyj, 2024.)

Tekoälyn avulla voidaan automatisoida rutiinitehtäviä, kuten tiedon syöttämistä, laskelmien tekemistä sekä vaativien kaaviokuvien tarkastamista ja niiden päivittämistä. Automatisoinnin avulla voidaan nopeuttaa tarjousprosessia sekä vapautta työntekijöiden aikaa korkeampaa asiantuntemusta vaativampiin tehtäviin. (Salo, 2024; Admicom Oyj, 2024.)

Tekoälyalgoritmit voivat optimoida tarjouslaskennan kokonaisuutta dynaamisesti huomioiden raaka-aineet, työvaiheet, koneiden kapasiteetin, aikataulut, katerakenteen ja resurssien allokoinnin. Tämä on erityisen hyödyllistä suurissa ja monimutkaisissa projekteissa, joissa hinnoittelun ja kustannusten hallinta manuaalisesti on haastavaa. (Salo, 2024.)

Tekoälyä hyödyntävät järjestelmät voivat analysoida rakennuspiirustuksia tehokkaasti käyttäen konenäköä ja luonnollisen kielen käsittelyä. Ne tunnistavat automaattisesti piirustuksista ja asiakirjoista elementtejä, kuten sähkökaapeleita, seiniä, putkistoja, sekä laskevat tarvittavat materiaalimäärät. Nämä järjestelmät nopeuttavat ja tarkentavat laskentaprosessia sekä parantavat eri ammattilaisten välistä yhteistyötä järjestelmän tuottamalla yhdenmukaisella datalla. (Admicom Oyj, 2024.)

3.7.2 Tekoäly osana hankintaprosessia

Tekoälyllä voidaan tehostaa hankintaprosessia. Sen avulla yritys voisi valita parhaat toimittajat, koska tekoäly pystyy analysoimaan toimittajien

suorituskykyä ja luotettavuutta aikaisempien tietojen perusteella. Tekoäly voi vähentää sopimusrikkomusten riskiä analysoimalla sopimusehtoja ja varmistamalla, että sopimusehdot noudattavat yrityksen politiikkaa ja säädöksiä. Kustannusten hallinta voi myös helpottua tekoälyn avulla. Tekoäly voi seurata markkinahintoja ja ennustaa hintojen muutoksia, joiden avulla yritys pystyisi tekemään kustannustehokkaita hankintapäätöksiä. Tekoälymallit voivat vähentää yllättäviä kustannuksia ja tunnistaa piilotettuja kustannuksia. Tekoäly voi jopa ehdottaa säästöjä tuovia, vaihtoehtoisia ratkaisuja. (Salo, 2024.)

3.8 Tekoälyn haasteet tarjouslaskennassa

Tekoäly toimii tehokkaasti vain tarkoin rajatuissa ja toistuvissa tehtävissä, joihin se on opetettu. Sen suorituskyky on rajallinen monimutkaisten ja moniulotteisten ongelmien ratkaisemisessa. Rajallisuus tulee esiin erityisesti silloin, kun dataa on niukasti tai ongelma ei ole selkeästi määriteltävissä. Yritykset saattavat odottaa tekoälyltä ratkaisuja haastaviin ongelmiin, vaikka reaalimaailmassa ihanteellinen lopputulos ei ole saavutettavissa. (Kananen, H. & Puolitaival, 2019, s.17–18, 40.)

Jotta tekoälyä voitaisiin hyödyntää, se vaatii tietämyksen olevassa olevista teknologioista, uusien järjestelmien käyttöönoton tuomia kustannuksia sekä henkilöstön koulutusta. Koska tekoäly käsittelee suuria määriä hankkeisiin liittyvää tietoa, on tietoturvariskit otettava huomioon. (Ala-Rakkola, 2020, s. 1; Admicom Oyj, 2024.)

Määrälaskenta on siirtymässä kohti ennakoivaa analytiikkaa ja reaaliaikaisia päivityksiä, mikä muuttaa merkittävästi alan toimintatapoja. Tekoäly ja automaatio mahdollistavat nopeamman ja tarkemman määrälaskennan ja tämä kehitys tarjoaa merkittävää kilpailuetua yrityksille, jotka hyödyntävät uusia teknologioita eturintamassa. Tämä voi tuoda haasteita perinteisille yrityksille, jotka eivät ole vielä siirtyneet uusiin määrälaskentamenetelmiin. (Admicom Oyj, 2024.)

4 TEKOÄLYPOHJAISET TYÖKALUT JA MENETELMÄT TARJOUSLASKENNASSA

4.1 Käytetyt ohjelmat

Työssä käytettiin OpenCV-ohjelmakirjastoa. OpenCV on lyhenne sanoista Open Source Computer Vision Library. Se on suosittu avoimen lähdekoodin kirjasto, joka sisältää laajan valikoivan algoritmeja kuvankäsittelyyn, objektien tunnistukseen, koneoppimiseen ja moneen muuhun tehtävään. OpenCV tukee useita ohjelmointikieliä kuten Pythonia, C++:a ja Javaa. OpenCV toimii monilla alustoilla kuten Windows, macOS, Linux ja Android. (OpenCV,2025).

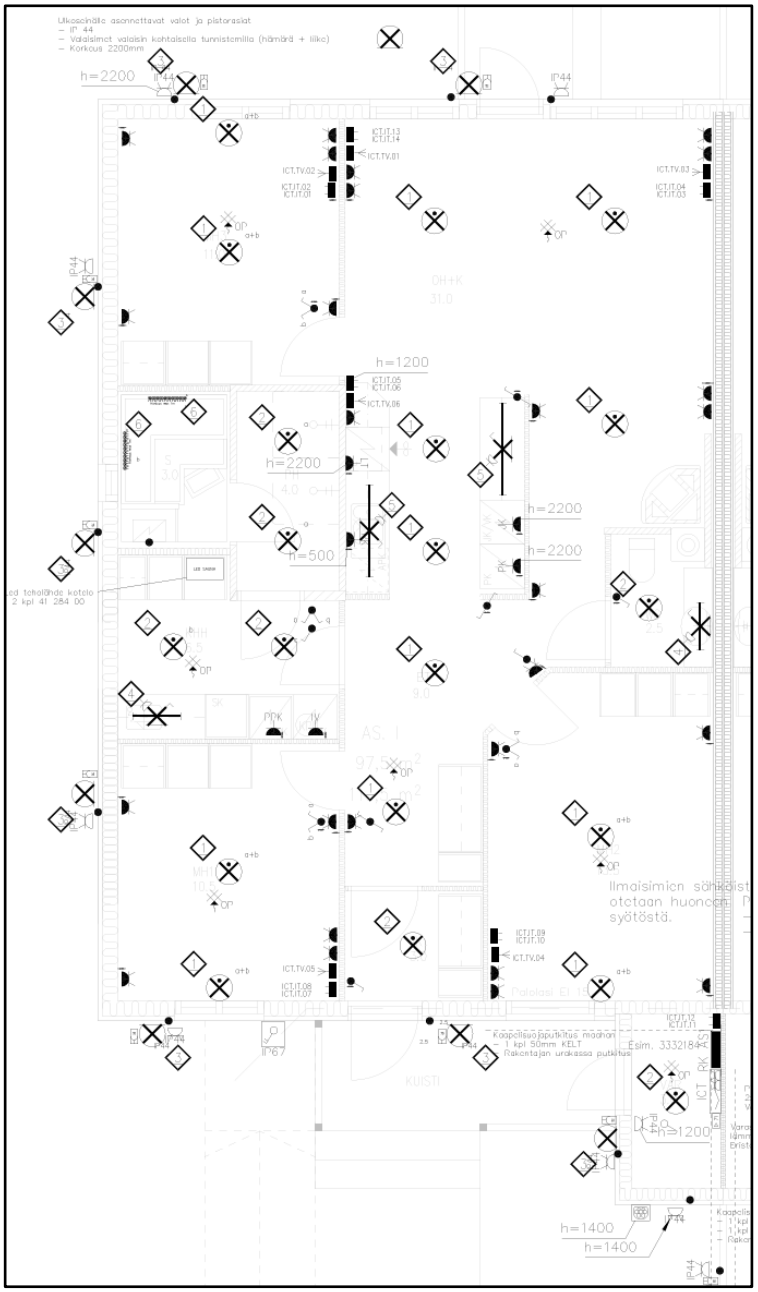
Opinnäytetyössä hyödynnettiin tekoälypalvelua ja ohjelmointi tehtiin Python-ohjelmointikielellä. Ohjelmakoodin kirjoittamiseen käytettiin Visual Studio Code-a. Ohjelmalla haluttiin etsiä eri komponentteja pistepiirustuksesta kuvantunnistuksen avulla. Tavoitteena oli tehdä koodi, joka tunnistaa halutun komponentin symbolin piirikaaviosta, laskee komponenttien määrän, merkitsee löytämänsä symbolit tulostekuvaan ja näyttää tulostekuvan käyttäjälle.

4.2 Ohjelman tekeminen

Ohjelma aloitettiin tuomalla tarvittavat moduulit (kuva 2). `Cv2` määrittää, että OpenCV:tä käytetään kuvan käsittelyyn ja `numpy`-kirjastoa tarvitaan kuvien skaalaamiseen sekä matemaattisten operaatioiden käsittelyyn. Matplotlib-kirjaston `pyplot`-moduulia käytetään tulosten esittämiseen graafisessa muodossa. Ohjelman pohjana toimi pistepiirustus (kuva 3), josta etsittiin tiettyjä symboleita, kuten kattovalaisimen symbolia.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Kuva 2. Moduulien tuominen



Kuva 3. Pistepiirustus

4.3 Kattovalaisimen symbolin etsiminen

Pistepiirustuksesta otettiin kattovalaisimen symbolin mallikuva (kuva 4). Ohjelman tehtävä oli etsiä pistepiirustuksesta valaisimen symbolia.

Kuva 4. Kattovalaisimen symbolin mallikuva

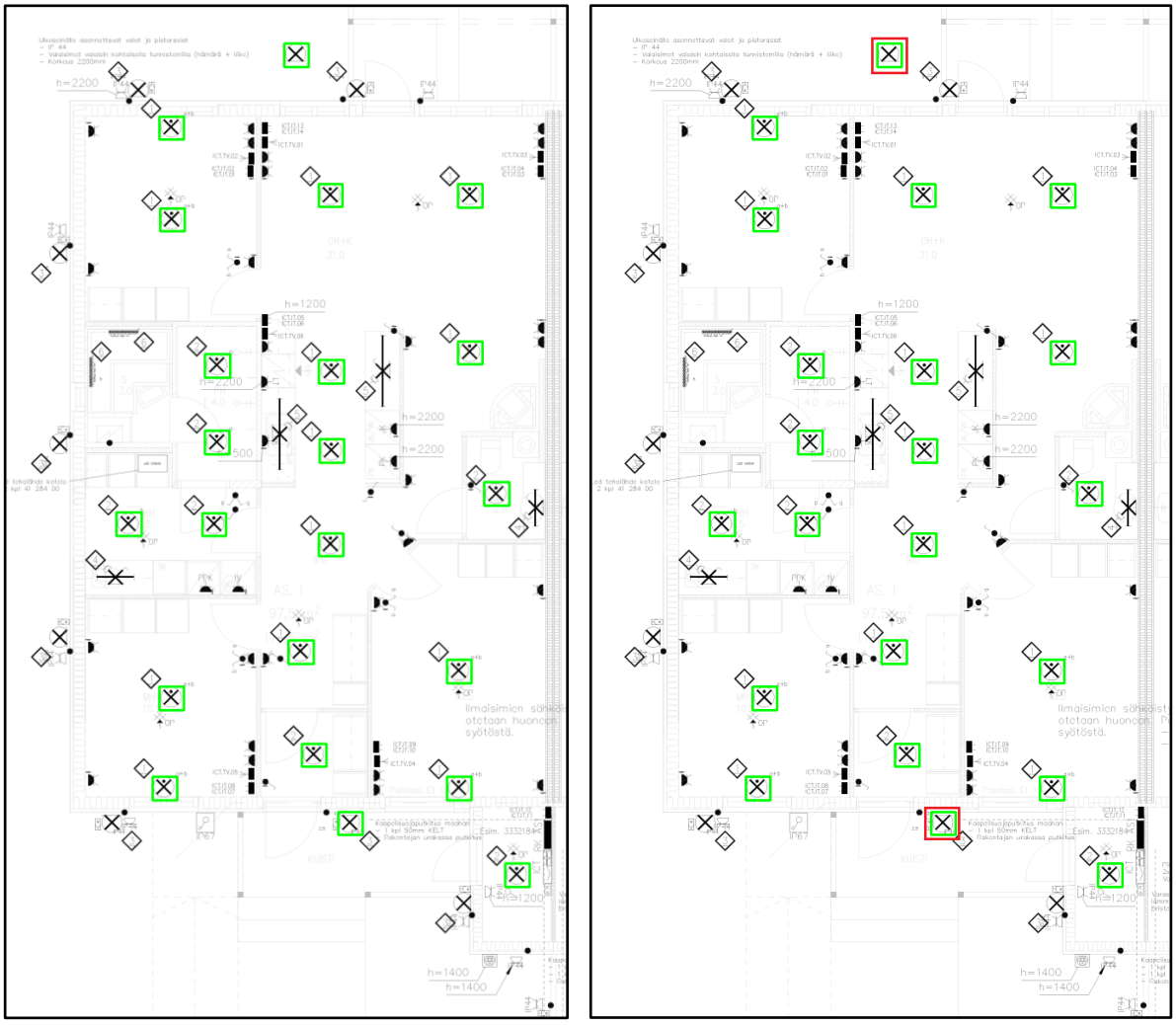
Pääohjelmaksi luotiin find_lamp_symbols_with_scaling-funktio (liite 1). Aluksi ladattiin lähdekuvaksi pistepiirustus (kuva 3), mallikuvaksi kattovalaisimen symboli (kuva 4) sekä tulostuskuva. Ohjelmaan määritettiin muuttujat `rectangles` ja `scales`. Rectangles on lista, johon tallennetaan tunnistettujen symbolien sijainnit suorakulmioina ja scales tallentaa käytetyt skaalat, jos niitä tarvitaan analyysiin.

Tämän jälkeen mallikuvaa skaalataan `np.linspace`-funktiolla eri kokoon välillä 50 % - 150 % alkuperäisestä, jotta valaisimen symbolit voidaan havaita riippumatta siitä, ovatko ne suurempia tai pienempiä kuin alkuperäinen mallikuva. Funktiolla `cv2.matchTemplate` skaalattu malli sovitetaan lähdekuvaan (kuva 3) ja tämä funktio palauttaa vastaavuustulokset, jotka kuvaavat, kuinka hyvin skaalattu malli vastaa lähdekuvan sisältöä.

Seuraavaksi määritettiin `threshold`-arvo eli kynnysarvo. Tämä arvo määrittää, kuinka hyvin mallikuvan ja lähdekuvan vastaa toisiaan. Tämä arvo on välillä -1 ja 1, missä 1 tarkoittaa täydellistä vastaavuutta, -1 tarkoittaa täydellistä negatiivista korrelaatiota ja 0 tarkoittaa, ettei vastaavuutta ole. Koodissa on käytetty threshold-arvoa 0,8. Tämä tarkoittaa, että mallin ja kuvan välinen kynnysarvo on oltava vähintään 0,8, jotta mallia pidettäisiin onnistuneena tunnistuksena. Jos threshold-arvoa korotetaan, ohjelma hyväksyy hyvin tarkat vastaavuudet. Tällöin väärien tunnistusten määrä vähenee, mutta herkkyys pienenee. Jos taas threshold-arvoa lasketaan, ohjelma hyväksyy vähemmän tarkkoja vastaavuuksia. Tämä taas lisää symbolin löytämisen mahdollisuutta, mutta samalla väärien tunnistusten määrä kasvaa. Ohjelmaa kokeiltiin eri treshold-arvoilla: esimerkiksi arvolla 0,7 saatiin viisi väärää tunnistusta ja arvolla 0,9 ohjelma ei löytänyt yhtään valaisinta.

Löydettyjen symboleiden sijainnit ja mitat tallennetaan `rectangles`-listaan ja käytetty skaalaustaso `scales`-listaan. Etsitty symboli voi löytyä useita kertoja eri skaalauksilla. Nämä päällekkäiset tunnistukset suodatetaan `cv2.groupRectangles`-funktiolla, joka yhdistää päällekkäiset tunnistukset yhdeksi suorakulmioksi. Jotta tulosten analysoiminen olisi helpompaa, suodatetut suorakulmiot piirretään lähdekuvaan vihreillä reunuksilla käyttämällä `cv2.rectangle`-funktiota. Tämä tulostekuva tallennetaan tiedostoon nimeltä `output_result_scaled_filtered.png`. Tuloksena saatu kuva näytetään käyttäjälle käyttämällä `matplotlib.pyplot`-käskyä. Löydettyjen symbolien määrä lasketaan suorakulmioiden lukumäärän perusteella ja tämä määrä tulostetaan käyttäjälle.

Ohjelma löytää 22 kattovalaisimen symbolia ja tekee tulostekuvan (kuva 5). Kattovalaisimia on alkuperäisessä lähdekuvassa 20 eli ohjelma tulkitsee kaksi kattovalaisinta väärin. Nämä kaksi valaisinta ovat seinävalaisimia ja niiden symboli on hieman erilainen, kuin etsityn valaisimen symboli. Seinävalaisimen symbolissa on yksi suora sivu eikä siinä ole jakorasiaa ilmaisevaa ympyrää.



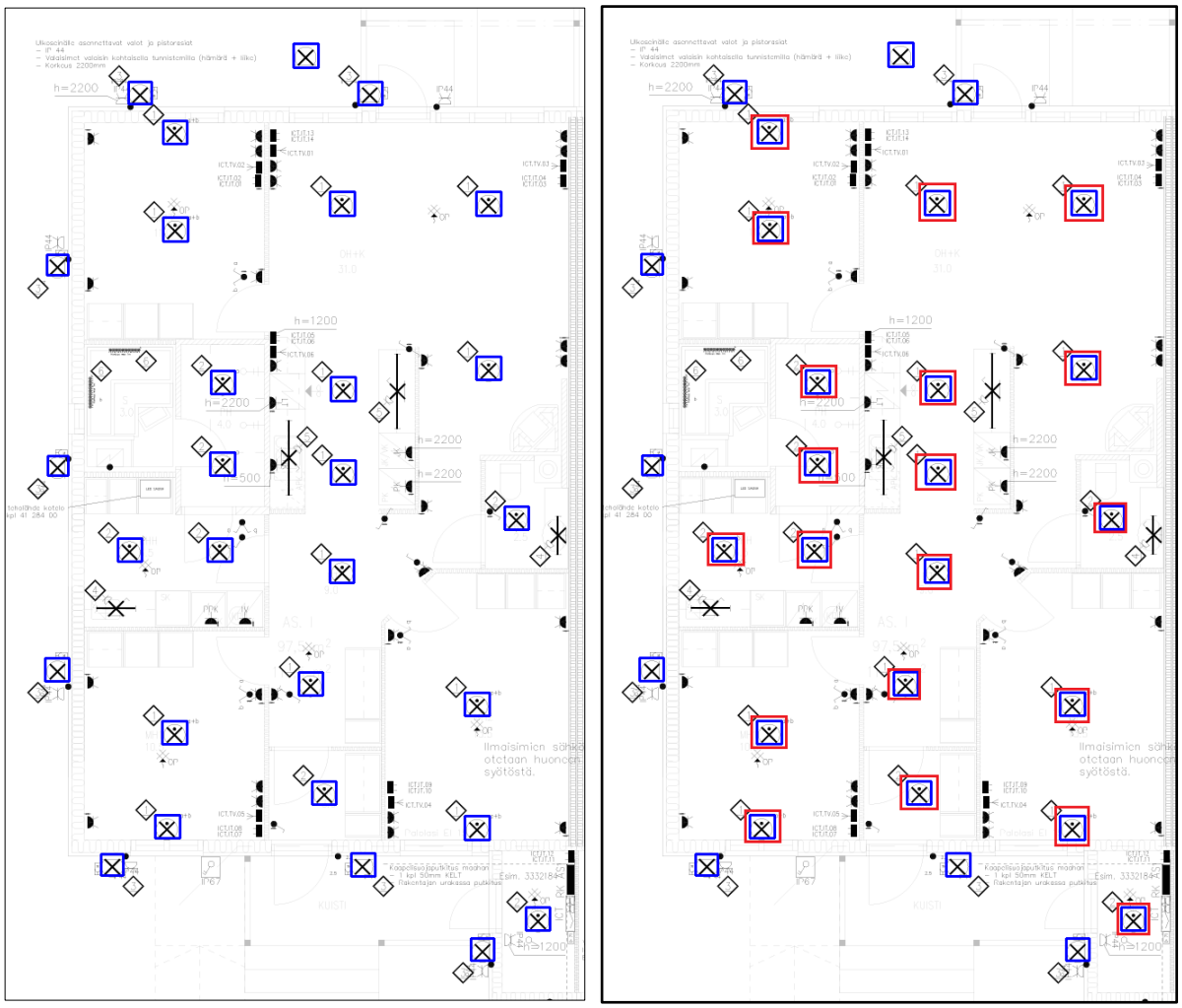
Kuva 5. Vasemmalla tulostekuva löydetyistä kattovalaisimista. Oikealla olevaan kuvaan merkitty väärät tunnistukset punaisella.

4.4 Seinävalaisimen symbolin etsiminen

Seuraavaksi kokeiltiin löytää pistepiirustuksesta seinävalaisimen symbolit (kuva 6). Pääohjelma pidettiin muuten samana, kuin kattolampun kohdalla, mutta mallikuva muutettiin vastaamaan seinävalaisimen symbolia ja tulostuskuvaan suorakulmiot muutettiin sinisiksi (liite 2).

Kuva 6. Seinävalaisimen symboli.

Ohjelma löysi 29 seinävalaisimen symbolia (kuva 7). Oikea seinävalaisinten määrä on 9. Nyt ohjelma tulkitsi myös kaikki kattovalaisimet seinävalaisimiksi. Parhaimmaksi treshold-arvoksi osoittautui 0,75. Arvolla 0,65 ohjelma löysi kaikki seinävalaisimet, mutta virheellisiä tunnistuksia tuli 27 kappaletta. Arvolla 0,85 ohjelma löysi 7 seinävalaisinta ja teki 13 virheellistä tunnistusta.



Kuva 7. Vasemmalla tulostekuva löydetyistä seinävalaisimista. Oikealla olevaan kuvaan merkitty väärät tunnistukset punaisella.

4.5 Katto- ja seinävalaisimen symbolien etsiminen

Seuraavaksi tehtiin ohjelma (liite 3), jossa yhdistettiin samaan koodiin sekä kattovalaisimen että seinävalaisimen symbolin etsintä. Ohjelmaan lisättiin toiminto, joka vertaa löydettyjen suorakulmioiden sijainteja toisiinsa ja suodattaa

päällekkäiset suorakulmiot. Prioriteetiksi asetettiin kattovalaisimet eli ne käsiteltiin ensin ja seinävalaisimet suodatettiin kattovalaisimien perusteella.

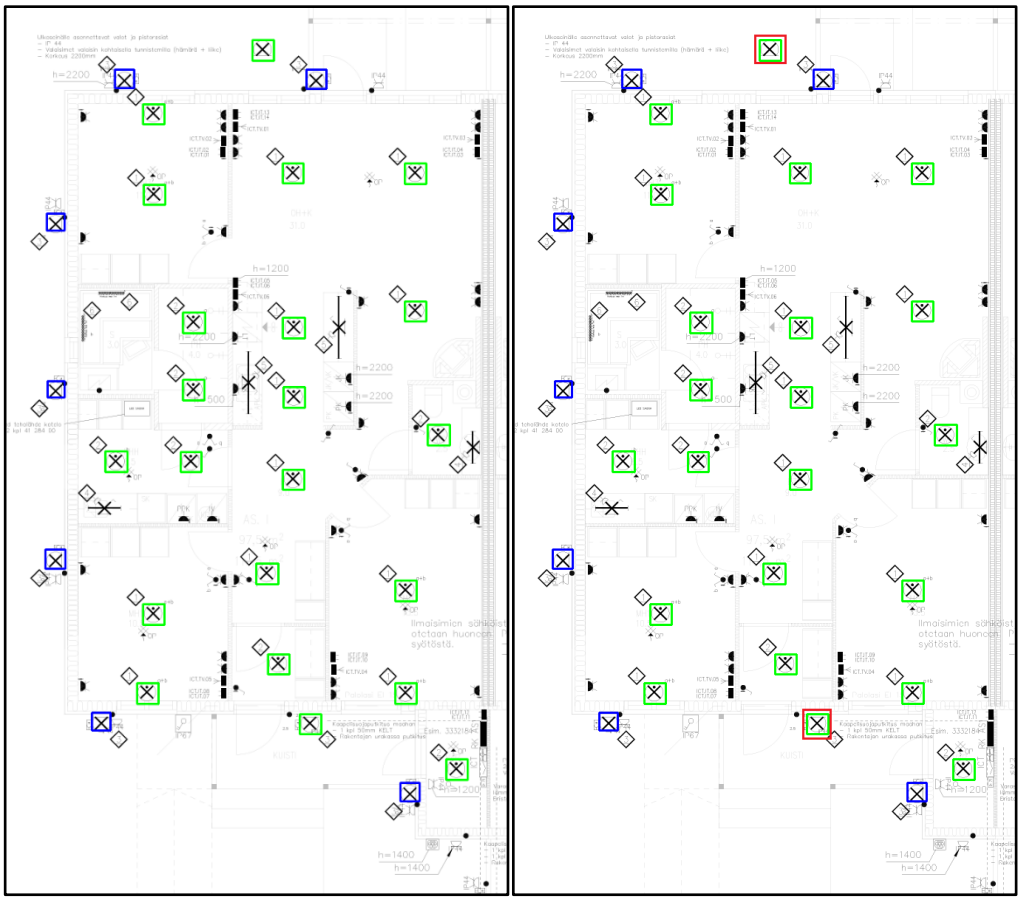
Funktio `find_and_filter_symbols` tunnistaa valaisinten symbolit eri malikuvien avulla ja poistaa päällekkäisyydet symbolien väliltä. Alustetaan lista `detected_symbols`, johon tallennetaan kaikkiin symboleihin liittyvät suorakulmiot. Tämän jälkeen iteroidaan jokainen mallikuva ja sen vastaava kynnysarvo erikseen sekä alustetaan `rectangles`-lista mallikuvan havaituille suorakulmioille. Mallikuva skaalataan ja suoritetaan vastaavuus lähdekuvan ja mallikuvan välillä kuten edellä. Koodiin lisättiin suorakulmioiden sijainnit ja nämä tallennettiin `rectangles`-listaan. Tämän jälkeen suodatetaan päällekkäiset suorakulmiot käyttämällä `groupRectangles`-funktiota. If-lauseessa määritellään, että jos käsitellään useampaa symbolia, suodatetaan suorakulmiot, jotka ovat päällekkäisiä aiempien symbolien kanssa. Etsityt symbolit asetetaan ns. tärkeysjärjestykseen. Nyt kattovalaisimet ovat ensisijaiset symbolit ja seinävalaisimet ovat toissijaiset. Ohjelma lisää suodatetut suorakulmiot `detected_symbols`-listaan ja palauttaa kaikkien symbolityyppien suorakulmiot.

Funktio `is_overlapping` määrittelee suorakulmioiden koordinaatit ja koot sekä tarkastaa, ovatko suorakulmiot erillisiä vai ovatko ne päällekkäisiä.

Ohjelma piirtää suorakulmiot lähdekuvaan `visualize_and_save`-funktiolla. Se piirtää vihreän suorakulmion kattovalaisimelle ja sinisen seinävalaisimelle. Funktio tulostaa käyttäjälle kunkin symbolityypin ja löydettyjen symbolien lukumäärän. Tulostekuva tallennetaan ja se näytetään käyttäjälle graafisesti.

Pääohjelmassa määritetään lähdekuvan sijainti, mallikuvat, kynnysarvot ja värit eri symboleille. Se tunnistaa symbolit mallikuvien avulla ja suodattaa päällekkäisyydet. Tulostekuvaan piirtyy suorakulmiot etsittyjen symbolien ympärille ja tulostekuva näytetään käyttäjälle.

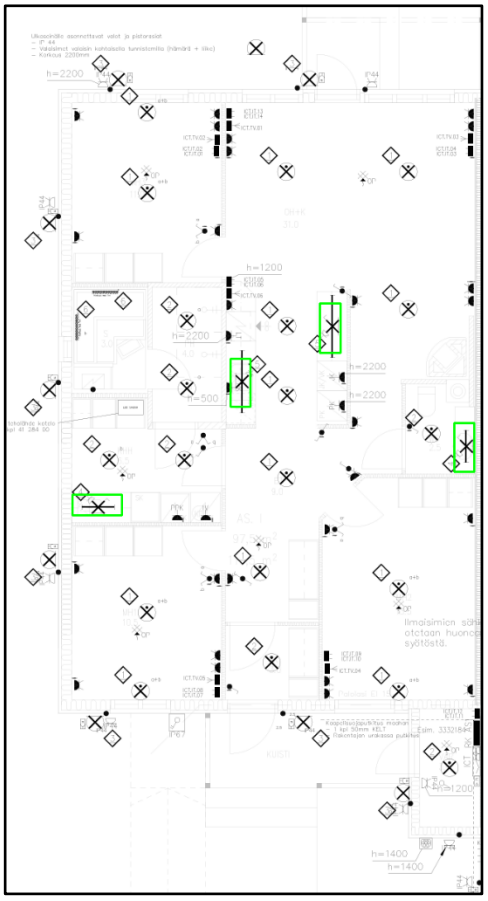
Ohjelma löytää 22 kattovalaisinta ja 7 seinävalaisinta (kuva 8). Oikeat määrät ovat 20 kattovalaisinta ja 9 seinävalaisinta. Ohjelma tulkitsee kaksi seinävalaisinta kattovalaisimiksi, eikä tätä saatu korjattua treshold-arvoja muuttamalla.



Kuva 8. Vasemmalla tulostekuva löydetyistä valaisimista. Oikealla olevaan kuvaan merkitty väärät tunnistukset punaisilla suorakulmioilla.

4.6 Loisteputkivalaisimen etsiminen

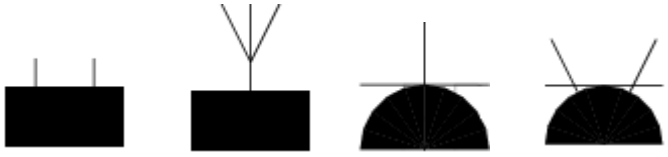
Lähdekuvassa (kuva 3) loisteputkivalaisimen symboli on piirretty joko pysty- tai vaakasuunnassa. Ohjelman perusidea pidetään samanlaisena kuin muidenkin valaisinten kohdalla eli ohjelma etsii annetun komponentin symbolia lähdekuvasta. Ohjelmaan lisättiin mallikuvan rotaatio eli kuvan kääntäminen. Tämä onnistuu `rotated_templates`-funktiolla, joka kääntää mallikuvaa 90 astetta. Näin ohjelma (liite 4) voi löytää loisteputkivalasimen symbolin sekä pysty- että vaakasuunnassa. Ohjelma löytää kaikki neljä loisteputkivalaisinta lähdekuvasta treshold-arvolla 0,65 eikä virheellistä tunnistusta tule (kuva 9).



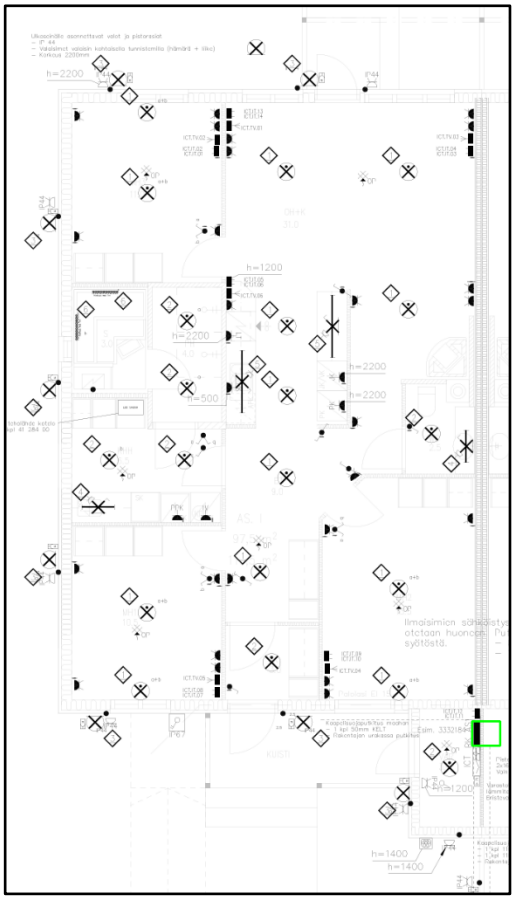
Kuva 9. Tulostekuva löydetyistä loisteputkivalaisimista.

4.7 Erilaisten pistorasioiden etsiminen

Lähdekuvassa on neljä erilaista pistorasiasymbolia: atk-pistorasia, antennipistorasia, sekä yksi- ja kaksiosainen suojakosketinpistorasia (kuva 10). Kun ohjelma rakennettiin samalla tavalla kuin lamppujen osalta, niin ohjelma löysi ainoastaan yhden pistorasian symbolin ja tämäkin oli väärä tunnistus (kuva 11). Tämänkaltainen ohjelma ei siis toimi pistorasioita etsittäessä.



Kuva 10. Atk-pistorasian, antennipistorasian, sekä yksi- ja kaksiosaisen suojakosketinpistorasian symbolit.



Kuva 11. Tulostekuva löydetyistä pistorasioista.

5 JOHTOPÄÄTÖKSET

Työssä käytetty OpenCV-kirjasto ja Template Matching-menetelmä sopi komponenttien tunnistukseen, kun etsittävän symbolin muoto tai ulkonäkö ei vaihdu eli kun mallikuva on lähes identtinen etsittävien symbolien kanssa. Symbolien tunnistuksessa käytettiin mallikuvien skaalaamista ja rotaatiota sekä kynnysarvon säätämistä. Mallikuvat skaalattiin, jotta ohjelma pystyisi tunnistamaan eri kokoiset symbolit. Rotaation avulla symbolit voitiin tunnistaa sekä pysty- että vaakasuunnassa. Ohjelmassa voitiin säätää kynnysarvoa eli sitä, kuinka hyvin mallikuva ja lähdekuva vastaa toisiaan.

Taulukossa 6 on esitetty, miten kynnysarvon säätäminen vaikutti tunnistuksen herkkyyteen ja vihreällä on ilmaistu tarkin tulos. Oikean kynnysarvon

löytäminen vaati manuaalista tarkastamista jokaisen symbolin kohdalla erikseen, joten ohjelma ei toiminut automaattisesti parhaalla mahdollisella tavalla. Liian korkea kynnysarvo jätti osan symboleista tunnistamatta, kun taas alhaisempi arvo johti useampiin virheellisiin tunnistuksiin.

Taulukko 6. Katto-, seinä- ja loisteputkivalaisinten tunnistaminen eri kynnysarvoilla erikseen testattuna

	Kynnysarvo	**Todellinen valaisinten määrä**	**Tunnistettujen valaisinten määrä**	**Oikein tunnistettujen valaisinten määrä**	**Löytämättä jääneiden valaisinten määrä**	**Virheellisesti tunnistettujen valaisinten määrä**
Katto-valaisin	0,6	20	32	20	0	12
	0,7	20	25	20	0	5
	0,8	20	22	20	0	2
	0,9	20	0	0	20	0
Seinä-valaisin	0,65	9	36	9	0	27
	0,75	9	29	9	0	20
	0,85	9	22	7	2	13
	0,9	9	3	3	6	0
Loiste-putki-valaisin	0,55	4	26	4	0	22
	0,65	4	4	4	0	0
	0,75	4	3	3	1	0
	0,85	4	1	1	3	0

Menetelmä soveltui parhaiten kattovalaisimien sekä loisteputkivalaisimien symbolien tunnistamiseen. Seinävalaisinten kohdalla ohjelma tulkitsi sekä katto- että seinävalaisimet seinävalaisimiksi. Kun ohjelma toteutettiin niin, että se tunnisti molemmat valaisintyypit samanaikaisesti, valaisimien erottelutarkkuus parani selvästi. Tässäkin ohjelmassa sopiva kynnysarvo piti manuaalisesti kokeilla (taulukko 7).

Taulukko 7. Katto- ja seinävalaisimien tunnistaminen yhteisellä ohjelmalla

Katto-valaisimen kynnys-arvo	Seinä-valaisimen kynnys-arvo	Etsityn valaisimen tyyppi	Todellinen valaisinten määrä	Tunnistettujen valaisinten määrä	Oikein tunnistettujen valaisinten määrä	Löytämättä jääneiden valaisinten määrä	Virheellisesti tunnistettujen valaisinten määrä
0,7	0,7	Katto	20	25	20	0	5
		Seinä	9	8	4	0	4
0,8	0,75	Katto	20	22	20	0	2
		Seinä	9	7	7	2	2
0,9	0,85	Katto	20	0	0	20	0
		Seinä	9	22	7	2	15

Erityisesti pistorasioiden symbolien tunnistus epäonnistui lähes täysin. Tämä johtuu siitä, että pistorasioiden symbolit ovat rakenteeltaan yksinkertaisia ja hyvin samankaltaisia, eikä Template Matching -menetelmä siksi kykene luotettavasti erottamaan eri tyyppejä toisistaan. Koska menetelmä ei tunnista symbolien pieniä yksityiskohtaisia eroja, se ei ole paras mahdollinen työkalu tämäntyyppiseen kuvantunnistukseen.

Työssä käytetty kuvantunnistusratkaisu osoitti, että Template Matching -menetelmä voisi oikein käytettynä auttaa sähköurakoitsijaa tarjouslaskennassa. Jotta työssä kehitettyä ohjelmaa voitaisiin hyödyntää laajemmin tarjouslaskennan tukena, sen tarkkuutta tulisi kuitenkin parantaa. Menetelmän suurimmat haasteet liittyivät samankaltaisten symbolien luotettavaan erotteluun sekä yksinkertaisten symbolien tunnistamiseen. Lisäksi esiintyneiden virheiden määrä osoittaa, että menetelmä nykyisellään vaatisi vielä manuaalista tarkastusta ennen kuin ohjelmasta saatua tietoa voitaisiin käyttää tarvikkeiden määriä laskettaessa ja hinnoittelussa.

Kuvantunnistuksen luotettavuutta voitaisiin jatkossa parantaa merkittävästi hyödyntämällä kehittyneempiä tekoälypohjaisia menetelmiä, kuten syväoppivia neuroverkkoja. Neuroverkot soveltuvat erityisen hyvin kuvantunnistukseen, sillä ne pystyvät oppimaan tehokkaasti pieniäkin eroja symbolien välillä. Näiden kehittyneempien menetelmien avulla olisi mahdollista vähentää

merkittävästi manuaalista työtä sekä parantaa tarjouslaskennan laatua ja tehokkuutta huomattavasti enemmän kuin Template Matching -menetelmällä yksinään. Tulevaisuudessa neuroverkkoihin perustuvat ratkaisut voivatkin tarjota luotettavampia ja monipuolisempia mahdollisuuksia kuvantunnistukseen.

LÄHTEET

Admicom Oyj. (2.12.2024). Määrälaskenta uudistuu vuonna 2025 – oletko valmis?. Talotekniikka-lehti. https://talotekniikka-lehti.fi/maaralaskenta-uudistuu-vuonna-2025-oletko-valmis/

Ahonen, T., Aro, J., Asikainen, J., Billing, M., Christophe, F., Gautam, M., Haapakoski, T., Holamo, O., Kapiainen, P., Karvonen, H., Kolehmainen, P., Kytöharju, J., Lanz, M., Latokartano, J., Leinonen, J., Liljamo, J., Liuha, A., Närhi, J. … Skriko, T. (2023). Teollisuuden robotiikka. Suomen Robotiikkayhdistys ry.

Ala-Rakkola, J. (2020). Koneoppiminen osana tarjouslaskentaa [YAMK opinnäytetyö, LAB-ammattikorkeakoulu]. Theseus. https://urn.fi/URN:NBN:fi:amk-2020112123626

Alpaydin, E. (2021). Koneoppiminen. Terra Cognita.

Fry, H. (2019). Hello world: kuinka selviytyä algoritmien aikakaudella. Bazar.

Järvinen, P. (2023). Tekoäly ja minä. Tammi.

Kananen, H. & Puolitaival, H. (2019). Tekoäly – bisneksen uudet työkalut. Alma Talent Oy.

Merilehto, A. (2018). Tekoäly – matkaopas johtajalle. Alma Talent.

OpenCV. (2025). Vision AI: Image & Visual AI Tools | Google Cloud. https://docs.opencv.org/4.x/d1/dfb/intro.html

Saastamoinen, A. & Autio, I. (2017). Sähköurakoitsijan tarjouslaskenta. Sähköinfo Oy.

Salo, P. (27.11.2024). Tekoäly tehostamassa tarjouslaskentaa ja hankintaprosessia. Sofor. https://www.sofor.fi/news/blogi/digin-aitiopaikalta/tekoaly-tarjouslaskennassa/

Siukonen, T. & Neittaanmäki, P. (2019). Mitä tulisi tietää tekoälystä. Docendo.

Sähköinfo oy. Sähköurakan yksikkökustannuksia 2024/II Hinnoitteluohjeet. (2024). Sähköinfo oy. https://severi.sahkoinfo.fi

Sähköliitto. (2024). Urakkalaskentaesimerkit 2023–2025, päivitetty. Haettu 15.3.2025 osoitteesta https://sahkoliitto.fi/wp-content/uploads/2024/08/Urakkalaskentaesimerkit-2023-2025-PALTA-STTA-SAHKOLIITTOpaivitys-5.6.2024.pdf

Sähkötekniset työnantajat STTA ry, Palvelualojen työnantajat PALTA ry & Sähköalojen ammattiliitto ry. (2023). Sähköistys- ja sähköasennusalan työehtosopimus 2023–2025. Haettu 15.3.2025 osoitteesta https://sahkoliitto.fi/wp-content/uploads/2023/11/Sahkoistysalan_TES_2023.pdf

Taivainen, S., (2023). Rakennusurakoiden julkinen hankinta – pari sanaa urakkaneuvotteluista. https://ptcs.fi/rakennusurakoiden-julkinen-hankinta-pari-sanaa-urakkaneuvotteluista/

Toivanen, H., (20.3.2023). Mitä tekoäly on?. Kustannusosakeyhtiö Teos.

Vartiainen, H., Terde, M., Jormanainen, I., Kahila, J., Valtonen, T. & Toivonen, T. (2021). Tekoäly, koneoppiminen ja teknologian murros: Kohti datatoimijuutta ja tulevaisuuden design-taitoja. Ainedidaktiikka 5(2). https://doi.org/10.23988/ad.90776

LIITE 1: OHJELMAKOODI KATTOVALAISINTEN ETSIMISEEN

```python
def find_lamp_symbols_with_scaling(image_path, template_path, threshold):
    # Lataa piirikaavio ja lampun symboli mallikuvana
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE) # Lähdekuva
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE) # Mallikuva
    output_image = cv2.imread(image_path) # Tulostuskuva

    rectangles = []  # Tallennetaan kaikki löydetyt suorakulmiot
    scales = []  # Tallennetaan käytetyt skaalat, jos niitä tarvitaan analyysiin

    # Kokeillaan eri skaaloja
    for scale in np.linspace(0.5, 1.5, 20):  # Skaalaa välillä 50 % - 150 %
        resized_template = cv2.resize(template, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)
        result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
        locations = np.where(result >= threshold)

        for loc in zip(*locations[::-1]):
            h, w = resized_template.shape
            rectangles.append([loc[0], loc[1], w, h])  # Lisää suorakulmio koordinaatteineen
            scales.append(scale)  # Lisää käytetty skaala

    # Suodata päällekkäiset suorakulmiot
    rectangles, _ = cv2.groupRectangles(rectangles, groupThreshold=1, eps=0.5)

    # Piirretään vain suodatetut suorakulmiot ja lasketaan osumat
    lamp_count = len(rectangles)
    for (x, y, w, h) in rectangles:
        cv2.rectangle(output_image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Tulostetaan osumien määrä
    print(f"Löytyneiden kattovalaisinten määrä: {lamp_count}")

    # Tallenna tuloskuva tiedostoon
    output_result_path = 'C:/Opinnaytetyo_tiedostot/output_result_scaled_filtered.png'
    cv2.imwrite(output_result_path, output_image)
    print(f"Tuloskuva tallennettu: {output_result_path}")

    # Näytä kuva matplotlibilla
    plt.imshow(cv2.cvtColor(output_image, cv2.COLOR_BGR2RGB))
    plt.title("Detected Symbols")
    plt.axis('off')
    plt.show()

# Polut kuvalle ja mallikuvalle
image_path = r'C:/Opinnaytetyo_tiedostot/Pistepiirustus.png'  # Lähdekuva
template_path = r'C:/Opinnaytetyo_tiedostot/lamp_symbol.png'  # Mallikuva

# Kutsu funktiota
find_lamp_symbols_with_scaling(image_path, template_path, threshold=0.8)
```

LIITE 2: OHJELMAKOODI SEINÄVALAISINTEN ETSIMISEEN

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def find_lamp_symbols_with_scaling(image_path, template_path, threshold):
    # Lataa piirikaavio ja lampun symboli mallikuvana
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE) # Lähdekuva
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE) # Mallikuva
    output_image = cv2.imread(image_path) # Tulostuskuva

    rectangles = []  # Tallennetaan kaikki löydetyt suorakulmiot
    scales = []  # Tallennetaan käytetyt skaalat, jos niitä tarvitaan analyysiin

    # Kokeillaan eri skaaloja
    for scale in np.linspace(0.5, 1.5, 20):  # Skaalaa välillä 50 % - 150 %
        resized_template = cv2.resize(template, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)
        result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
        locations = np.where(result >= threshold)

        for loc in zip(*locations[::-1]):
            h, w = resized_template.shape
            rectangles.append([loc[0], loc[1], w, h])  # Lisää suorakulmio koordinaatteineen
            scales.append(scale)  # Lisää käytetty skaala

    # Suodata päällekkäiset suorakulmiot
    rectangles, _ = cv2.groupRectangles(rectangles, groupThreshold=1, eps=0.5)

    # Piirretään vain suodatetut suorakulmiot ja lasketaan osumat
    lamp_count = len(rectangles)
    for (x, y, w, h) in rectangles:
        cv2.rectangle(output_image, (x, y), (x + w, y + h), (255, 0, 0), 2)  # Sininen suorakulmio

    # Tulostetaan osumien määrä
    print(f"Löytyneiden seinävalaisimien määrä: {lamp_count}")

    # Tallenna tuloskuva tiedostoon
    output_result_path = 'C:/Opinnaytetyo_tiedostot/output_result_scaled_filtered_seinavalaisin.png'
    cv2.imwrite(output_result_path, output_image)
    print(f"Tuloskuva tallennettu: {output_result_path}")

    # Näytä kuva matplotlibilla
    plt.imshow(cv2.cvtColor(output_image, cv2.COLOR_BGR2RGB))
    plt.title("Detected Symbols")
    plt.axis('off')
    plt.show()

# Polut kuvalle ja mallikuvalle
image_path = r'C:/Opinnaytetyo_tiedostot/Pistepiirustus.png'  # Lähdekuva
template_path = r'C:/Opinnaytetyo_tiedostot/lamp_wall_symbol.png'  # Mallikuva

# Kutsu funktiota
find_lamp_symbols_with_scaling(image_path, template_path, threshold=0.75)
```

LIITE 3: OHJELMAKOODI KATTO- JA SEINÄVALAISINTEN ETSIMISEEN

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def find_and_filter_symbols(image_path, template_paths, thresholds):
    """
    Tunnistaa symbolit eri malleilla ja suodattaa päällekkäisyydet.
    """
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    detected_symbols = []

    for i, (template_path, threshold) in enumerate(zip(template_paths, thresholds)):
        template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
        rectangles = []

        # Skaalaa mallikuvaa ja etsi osumat
        for scale in np.linspace(0.5, 1.5, 20):
            resized_template = cv2.resize(template, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)
            result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
            locations = np.where(result >= threshold)

            for loc in zip(*locations[::-1]):
                h, w = resized_template.shape
                rectangles.append([loc[0], loc[1], w, h])

        # Suodata päällekkäiset suorakulmiot
        rectangles, _ = cv2.groupRectangles(rectangles, groupThreshold=1, eps=0.5)

        # Poista päällekkäisyydet aikaisempien symbolien kanssa
        if i > 0:
            prev_rects = [rect for symbols in detected_symbols for rect in symbols]
            rectangles = [r for r in rectangles if not any(is_overlapping(r, pr) for pr in prev_rects)]

        detected_symbols.append(rectangles)

    return detected_symbols

def is_overlapping(rect1, rect2):
    """
    Tarkistaa, ovatko kaksi suorakulmiota päällekkäisiä.
    """
    x1, y1, w1, h1 = rect1
    x2, y2, w2, h2 = rect2
    return not (x1 + w1 <= x2 or x2 + w2 <= x1 or y1 + h1 <= y2 or y2 + h2 <= y1)
```

```python

def visualize_and_save(image_path, detected_symbols, colors, output_path):
    """
    Visualisoi ja tallentaa tunnistetut symbolit.
    """
    output_image = cv2.imread(image_path)

    for symbols, color in zip(detected_symbols, colors):
        for (x, y, w, h) in symbols:
            cv2.rectangle(output_image, (x, y), (x + w, y + h), color, 2)

    # Tulosta symbolien määrät
    for i, symbols in enumerate(detected_symbols):
        print(f"Symbolityyppi {i + 1}: {len(symbols)} löydetty")

    # Tallenna ja näytä tulos
    cv2.imwrite(output_path, output_image)
    print(f"Tuloskuva tallennettu: {output_path}")
    plt.imshow(cv2.cvtColor(output_image, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.show()

# Polut ja parametrit
image_path = r'C:/Opinnaytetyo_tiedostot/Pistepiirustus.png'
template_paths = [
    r'C:/Opinnaytetyo_tiedostot/lamp_symbol.png',
    r'C:/Opinnaytetyo_tiedostot/lamp_wall_symbol.png'
]
thresholds = [0.8, 0.75]
colors = [(0, 255, 0), (255, 0, 0)]  # Vihreä kattovalaisin, sininen seinävalaisin

# Tunnista ja visualisoi symbolit
detected_symbols = find_and_filter_symbols(image_path, template_paths, thresholds)
visualize_and_save(image_path, detected_symbols, colors, 'C:/Opinnaytetyo_tiedostot/output_result_combined.png')
```

LIITE 4: LOISTEPUTKIVALAISINTEN ETSIMISEEN

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def find_lamp_symbols_with_scaling_and_rotation(image_path, template_path, threshold):
    # Lataa piirikaavio ja lampun symboli mallikuvana
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  # Lähdekuva
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)  # Mallikuva
    output_image = cv2.imread(image_path)  # Tulostuskuva

    rectangles = []  # Tallennetaan kaikki löydetyt suorakulmiot
    scales = []  # Tallennetaan käytetyt skaalat
    rotations = []  # Tallennetaan käytetyt rotaatiot

    # Generoidaan rotaatiot (0 ja 90 astetta)
    rotated_templates = [template, cv2.rotate(template, cv2.ROTATE_90_CLOCKWISE)]

    # Käydään läpi eri skaaloja ja rotaatioita
    for rotated_template in rotated_templates:
        for scale in np.linspace(0.5, 1.5, 20):  # Skaalaa välillä 50 % - 150 %
            resized_template = cv2.resize(rotated_template, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)
            result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
            locations = np.where(result >= threshold)

            for loc in zip(*locations[::-1]):
                h, w = resized_template.shape
                rectangles.append([loc[0], loc[1], w, h])  # Lisää suorakulmio koordinaatteineen
                scales.append(scale)  # Lisää käytetty skaala
                rotations.append(rotated_template is rotated_templates[1])  # True jos rotaatio 90 astetta

    # Suodata päällekkäiset suorakulmiot
    rectangles, _ = cv2.groupRectangles(rectangles, groupThreshold=1, eps=0.5)

    # Piirretään vain suodatetut suorakulmiot ja lasketaan osumat
    lamp_count = len(rectangles)
    for (x, y, w, h) in rectangles:
        cv2.rectangle(output_image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Tulostetaan osumien määrä
    print(f"Löytyneiden symbolien määrä: {lamp_count}")

    # Tallenna tuloskuva tiedostoon
    output_result_path = 'C:/Opinnaytetyo_tiedostot/output_result_scaled_rotated_filtered_loisteputkivalaisin.png'
    cv2.imwrite(output_result_path, output_image)
    print(f"Tuloskuva tallennettu: {output_result_path}")

    # Näytä kuva matplotlibilla
    plt.imshow(cv2.cvtColor(output_image, cv2.COLOR_BGR2RGB))
    plt.title("Detected Symbols")
    plt.axis('off')
    plt.show()

# Polut kuvalle ja mallikuvalle
image_path = r'C:/Opinnaytetyo_tiedostot/Pistepiirustus.png'  # Lähdekuva
template_path = r'C:/Opinnaytetyo_tiedostot/Loisteputkivalaisin.png'  # Mallikuva

# Kutsu funktiota
find_lamp_symbols_with_scaling_and_rotation(image_path, template_path, threshold=0.65)
```