25/26

PWS SolarML

8402, 8250

JIVRAJ SINGH, KEVIN KANG

MENEER DE GRAAF, MEVROUW ROOS

Inhoud

Voorwoord

Dit is ons profielwerkstuk over de toepassing van machine learning bij het efficiënter onderhouden van zonnepanelen. Tijdens ons onderzoek hebben we bekeken hoe algoritmes kunnen helpen bij het opsporen van vervuiling op zonnepanelen, zodat onderhoud gerichter en duurzamer kan worden uitgevoerd. Dit onderwerp past goed bij de huidige ontwikkelingen binnen duurzame energie en bij onze technische interesses.

We hebben dit onderwerp omdat zonnepanelen een steeds grotere rol spelen in de energietransitie, maar vervuiling nog steeds zorgt voor onnodig rendementsverlies. Door het onderhoud slimmer te organiseren met behulp van machine learning, kan dit verlies worden beperkt.

Graag willen we meneer De Graaf en mevrouw Roos bedanken voor hun begeleiding, feedback en ondersteuning tijdens dit project. Hun hulp en het meedenken in zowel het onderzoek als de praktische uitvoering hebben ons enorm geholpen om dit profielwerkstuk tot een goed resultaat te brengen. Ook willen we iedereen bedanken die ons op welke manier dan ook heeft geholpen of geïnspireerd.

Wij wensen u veel leesplezier.

Jivraj Singh & Kevin Kang

Inleiding

Sinds de uitvinding van het eerste zonnepaneel in 1954 is zonne-energie uitgegroeid tot een van de belangrijkste duurzame energiebronnen ter wereld. Zonnepanelen spelen een grote rol in de energietransitie en worden steeds vaker ingezet in grote zonneparken. Ondanks deze groei is er nog steeds een fundamenteel, praktisch probleem op het gebied van efficiëntie van de zonnepanelen.

De efficiëntie wordt namelijk beperkt, want na verloop van tijd raken zonnepanelen vervuild door stof, pollen en vogelpoep. Deze vervuiling zorgt ervoor dat minder zonlicht de zonnecellen bereikt, waardoor het rendement daalt. Om dit te voorkomen moeten zonnepanelen regelmatig worden schoongemaakt. In de praktijk gebeurt dit vaak handmatig of met relatief eenvoudige reinigingssystemen. Dit kost veel tijd, arbeid en geld, vooral bij grote zonneparken. Hierdoor wordt onderhoud soms minder vaak uitgevoerd dan eigenlijk nodig is, wat leidt tot veel, onnodig energieverlies.

Het gebruik van zonne-energie blijft doorgroeien en de behoefte aan slimmere en efficiëntere onderhoudsmethoden neemt alleen maar toe. Dankzij nieuwe technologieën zijn er mogelijkheden om dit proces te verbeteren. Een van deze technologieën is machine learning. Door gegevens te verzamelen over vervuiling, weersomstandigheden en energieopbrengst kan een systeem leren wanneer en hoe zonnepanelen het beste schoongemaakt kunnen worden. Hierdoor kan onderhoud gerichter worden uitgevoerd en kan verspilling van tijd, water en energie worden verminderd.

In dit profielwerkstuk wordt onderzocht hoe machine learning kan helpen om zonnepanelen sneller en efficiënter schoon te maken. Eerst wordt uitgelegd wat machine learning precies inhoudt, hoe het werkt en wat het verschil is tussen handmatig functies zoeken in data en het automatisch leren van patronen via deep learning.

Daarna kijken we naar de modellen die geschikt zijn voor beeldanalyse, zoals Convolutional Neural Networks en Siamese Neural Networks. Hierin komen onderwerpen aan bod zoals image embeddings, het vergelijken van beelden en leren met beperkte labels. Ook wordt besproken hoe je data moet voorbereiden, en hoe technieken zoals het verdelen van een beeld in kleinere stukjes (tiles) en het maken van heatmaps helpen om vuil op zonnepanelen te vinden.

Om dit in de praktijk te brengen, wordt het gebruik van computer vision-bibliotheken zoals OpenCV behandeld, en wordt gekeken hoe sensoren en simulatiesoftware kunnen helpen bij het testen van een robot die deze taken uitvoert.

Het doel van dit onderzoek is te laten zien hoe slimme automatisering kan bijdragen aan een sneller, efficiënter en duurzamer onderhoud van zonnepanelen, en hoe machine learning een rol kan spelen bij het bepalen waar en wanneer gereinigd moet worden.

Onderzoeksvraag & Hypothese

Onderzoeksvraag:

Hoe kunnen we "machine learning" gebruiken om zonnepanelen efficiënter schoon te maken?

Hypothese

Machine learning kan worden toegepast om het reinigen van zonnepanelen efficiënter te maken door een autonome reinigingsrobot visueel vervuiling te laten herkennen en op basis daarvan te bepalen waar reiniging nodig is.

Met behulp van machine learning is de robot in staat om:

1. Vervuiling op zonnepanelen te detecteren.
2. Een inschatting te maken van hoe ernstig de vervuiling is.
3. Onderscheid te maken tussen schone en vervuilde delen van een zonnepaneel.
4. Alleen die delen te reinigen waar daadwerkelijk vervuiling aanwezig is.

De verwachting is dat een machine learning gestuurde robot met minder energie en minder menselijke inzet minimaal 80 procent van het rendementsverlies door vervuiling kan herstellen.

Daarnaast zorgt het gebruik van een autonome robot voor het voordeel dat zonnepanelen op moeilijk bereikbare locaties, zoals op grote hoogte of onbewoonde gebieden, veilig en regelmatig gereinigd kunnen worden zonder directe menselijke tussenkomst.

Theoretische achtergrond

Wat is machinaal leren?

Machine learning is een vorm van computertechniek waarbij algoritmen worden gemaakt om dingen te leren zoals mensen dat doen. In plaats van dat een computer precies verteld wordt wat hij moet doen, leert hij zelf door gegevens uit zijn omgeving te bekijken. Machine learning wordt tegenwoordig bijna overal gebruikt, en heeft daardoor invloed op het dagelijks leven van miljarden mensen.

Er zijn heel veel toepassingen. Denk bijvoorbeeld aan patroonherkenning, computer vision, ruimtevaart, financiën en medische of biologische onderzoeken. In al deze gebieden wordt machine learning op verschillende momenten ingezet.

In de medische wereld, bijvoorbeeld bij de behandeling van kanker met ioniserende straling (radiotherapie), speelt machine learning een grote rol. Radiotherapie bestaat uit meerdere stappen, zoals doktersbezoeken, het maken van een behandelplan en het bepalen hoe een patiënt reageert op de straling. Machine learning helpt om deze processen sneller en nauwkeuriger te maken. Het wordt gebruikt bij het maken van behandelplannen, beeldgestuurde bestraling, het volgen van ademhalingsbewegingen, en bij het voorspellen van hoe goed de behandeling zal werken. (Murphy, sd)

In de financiële wereld wordt machine learning weer op andere manieren gebruikt, zoals voor het voorspellen van faillissementen, aandelenkoersen, olieprijzen en voor het beheren van portefeuilles. Het helpt ook bij het opsporen van witwaspraktijken en andere vormen van fraude. (Artificial intelligence and machine learning in finance: A bibliometric review, 2022)

Machine learning is tegenwoordig diep verweven met onze samenleving. Het zit in onze telefoons, auto’s, ziekenhuizen en banken. Veel dingen die we dagelijks gebruiken zijn op de een of ander manier afhankelijk van machine learning, waardoor het steeds belangrijker wordt in ons leven en onze toekomst.

Wat zijn de gevolgen van viezigheden op zonnepanelen?

Stof op zonnepanelen is een ingewikkeld probleem dat ontstaat door allerlei factoren uit de omgeving. Wind (richting en snelheid), temperatuur, luchtvervuiling, luchtvochtigheid en zelfs de ligging van een zonnepark spelen allemaal een rol. Het vuil dat op de panelen terechtkomt kan van alles zijn, zoals zand, aarde, klei, vogelpoep, pollen en bladeren. Dit zorgt ervoor dat er minder zonlicht bij de zonnecellen komt, waardoor de opbrengst daalt.

Figuur 1 stof ophoping op het oppervlak van zonnepanelen

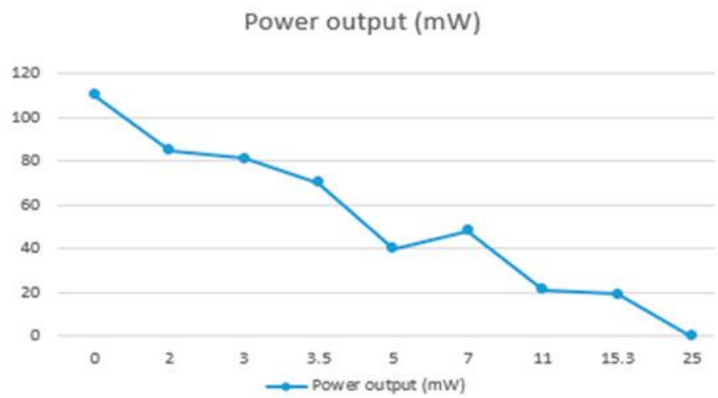
Daarnaast kunnen er schaduwplekken ontstaan en kan de temperatuur van het paneel omhooggaan.

Om te zorgen dat de panelen goed blijven werken, is het daarom belangrijk dat ze op de juiste manier worden schoongemaakt.

Experimenten zijn uitgevoerd om nauwkeuriger te kunnen bepalen hoe groot de invloed van zulke vervuiling is. Hieruit volgde dat een accumulatie van stof kan leiden tot een daling in het maximum opbrengst van de zonnepanelen van tientallen procenten, bijvoorbeeld 6% in een week en 20% per maand. Dit komt door de manier waarop zonlicht en stof interacteren. Een deel van de energie wordt geabsorbeerd door stof en omgezet in warmte-energie, terwijl een ander portie verspreid wordt over het oppervlak van de stof, alleen een klein fractie bereikte de onderliggende cellen in het paneel.

Als bladeren of vogelpoep een deel van een zonnepaneel blokkeren, daalt de opbrengst. Wanneer panelen in serie zijn aangesloten, verlaagt één zwak (beschaduwd) paneel de prestaties van alle panelen in de reeks. Daarom zijn er elektrische componenten parallel geschakeld met substrings van zonnecellen om vermogensverlies en schade door gedeeltelijke schaduw of defecte cellen te voorkomen. Deze componenten heten bypass-diodes. Ze leiden de stroom om de beschaduwde cellen heen zodat het systeem blijft werken, maar de totale opbrengst gaat alsnog omlaag.

De regio is dus erg van belang, in droge woestijngebieden hopen stof en vuil zich snel op op zonnepanelen. Dit kan een dikke laag vormen die het zonlicht blokkeert en zo de opbrengst aanzienlijk verlaagt. Afhankelijk van de hoeveelheid stof kan de efficiëntie binnen korte tijd tot circa 20–35% dalen. Hierdoor neemt de elektriciteitsproductie sterk af.

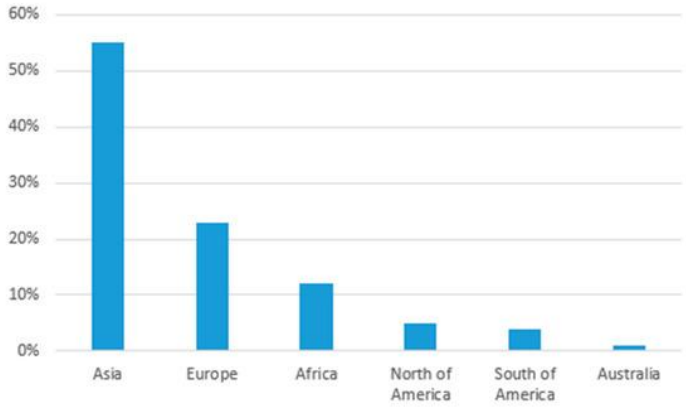


Figuur 2 Het effect van stof ophoping op elektrische energieproductie van zonnepanelen

De meeste onderzoeken naar stofophoping en vervuiling op zonnepanelen komen uit Azië. Vooral Saudi-Arabië loopt hierin voorop. Dat komt doordat:

- Ongeveer 60% van alle nieuwe zonnepanelen wereldwijd in Azië wordt geplaatst.
- Grote delen van Azië zijn droog, stoffig en winderig, waardoor er veel stof op zonnepanelen terechtkomt.

Afrika heeft ook veel gebieden waar veel zand en stof voorkomen.



Figuur 3 Contributie van het effect van stof ophoping op zonnepanelen in verschillende continenten

In kustgebieden kunnen zonnepanelen last krijgen van zout uit de zee. Dit kan op het oppervlak vast gaan zitten en zelfs corrosie veroorzaken, waardoor de panelen een heel stuk minder effectief worden. Soms kan het rendement zelfs met 40% dalen en zulke vuil is ook niet met alleen water schoon te maken.

In steden en industriële gebieden is de lucht vaak viezer door verkeer en fabrieken. Het vuil dat hierdoor op de zonnepanelen terechtkomt is vaak plakkerig en moeilijk schoon te maken, hiervoor geldt hetzelfde, alleen water of regen is niet genoeg om het schoon te maken.

In gebieden waar het vaak regent en de panelen schuin staan, zoals hier in Nederland, spoelt regen veel van het vuil ervan af.

Vuil op zonnepanelen heeft dus duidelijk een grote invloed op de opbrengst. In sommige gebieden kan dit het rendement zelfs met tientallen procenten verlagen. Omdat stof en andere viezigheid niet altijd vanzelf weggaat, is goede reiniging door middel van slimme techniek belangrijk om de panelen goed te laten werken. (Rachid, 2023)

Hoe worden zonnepanelen tegenwoordig schoon gemaakt?

Handmatig schoonmaken

Tegenwoordig bestaan er veel verschillende manieren om zonnepanelen schoon te maken. De eerste en meest simpele methode handmatig reinigen. Deze aanpak heeft helaas meerdere nadelen. Het levert namelijk maar weinig voordeel op als je kijkt naar de jaarlijkse opbrengst, omdat je niet dagelijks mensen eropuit kunt sturen om de panelen te reinigen. Daarnaast is het proces inefficiënt en brengt het risico's met zich mee: er wordt vaak veel water verspild en de mensen die het werk uitvoeren hebben soms te weinig kennis, wat schade aan de panelen kan veroorzaken.

Figuur 4 Schoonmaak proces van PV-zonnepanelen.

Watersproeiers (Water Spray Nozzle Systems)

Bij dit systeem worden zonnepanelen schoongespoeld met water dat via sproeikoppen over de panelen wordt verdeeld. Het water komt uit een ondergrondse tank die automatisch wordt aangevuld. Sensoren en slimme besturing regelen hoeveel water er nodig is. Dit systeem kan automatisch reinigen waardoor de opbrengst flink wordt verbeterd.

Figuur 5 Illustratie van een reinigingssysteem op verschillende zonnepanelen met sproeikoppen.

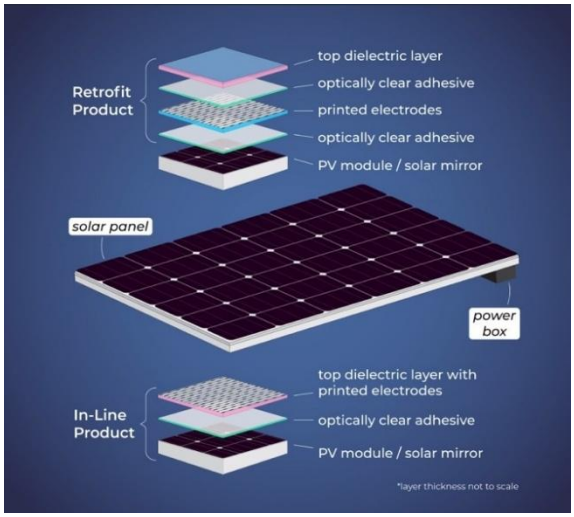
Reiniging met vrachtwagens

Voor grote zonneparken wordt vaak een truck met een borstel gebruikt. De truck rijdt tussen de rijen panelen door en maakt alles schoon. De druk van de borstel wordt gecontroleerd, zodat de panelen niet beschadigen. Het is een handige methode voor veel panelen tegelijk, maar je moet genoeg ruimte hebben en het blijft een beetje riskant als het terrein ongelijk is.

Figuur 6 Zonnepanelen reinigen met een vrachtwagen.

Elektrodynamisch scherm (EDS)

Een innovatieve manier is het EDS-systeem. Hierbij zit er een dunne laag met kleine elektroden op de panelen die het stof elektrisch wegduwen. Zo blijft het oppervlak schoon zonder dat er water of borstels nodig zijn. Het werkt ook op gebogen of lastig bereikbare panelen, waardoor ze langer optimaal blijven werken.



Figuur 7 EDS "Stack"

“Wash Panel” schoonmaakrobot

De Wash Panel-robot rijdt zelfstandig over de zonnepanelen en maakt ze schoon met een borstel en waterstraal. Hij kan op verschillende soorten daken en installaties worden gebruikt en kan zelfs op afstand gevolgd worden via een telefoon. Testen laten zien dat de opbrengst duidelijk stijgt, vooral bij panelen die eerst flink vuil waren.

Figuur 8 De "Wash Panel" schoonmaak robot.

Drones

Drones worden ook ingezet om panelen schoon te maken, vooral in grote of moeilijk bereikbare zonneparken. Ze blazen stof en vuil van de panelen met de krachtige luchtstroom van hun propellers. De drones volgen vooraf bepaalde routes zodat elk paneel wordt bereikt. Het werkt het beste in droge en stoffige gebieden en kan de energieproductie flink verbeteren.

Figuur 9 Drone gebruikt voor paneel reiniging.

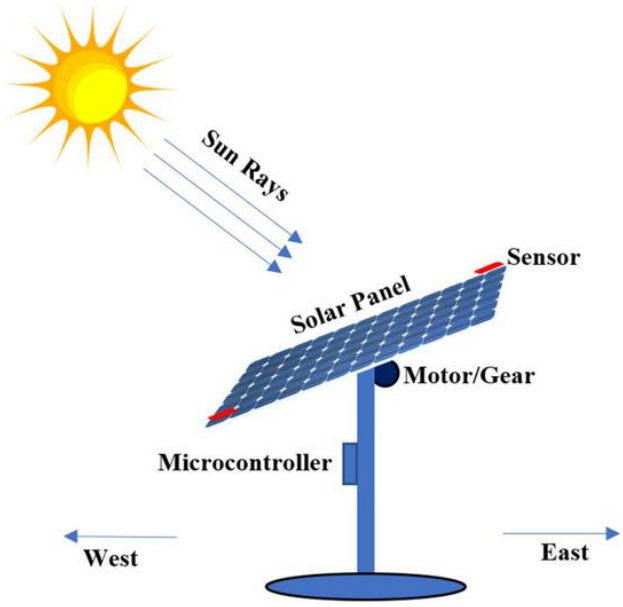
Resola-robot

De Resola-robot kan zelfstandig van paneel naar paneel rijden zonder rails of draden. Hij gebruikt een borstel, wiper en waterreservoir om het vuil te verwijderen en beschikt over sensoren die hem helpen obstakels te vermijden. Regelmatig gebruik van deze robot zorgt voor schonere panelen en een meetbare stijging in de energieopbrengst.

Figuur 10 Resola, PV-paneel schoonmaak robot.

Zelfreinigende & zonvolgende panelen

Er bestaan ook panelen die zichzelf schoonmaken en tegelijkertijd de zon volgen. Een wiper veegt het vuil weg terwijl motoren het paneel draaien zodat het altijd optimaal zonlicht opvangt. Sensoren meten de zon en sturen alles automatisch aan. Zo blijven de panelen schoon en leveren ze meer energie, zonder dat iemand er zelf iets voor hoeft te doen.



Figuur 11 Actief zonvolgsysteem.

Welke programmeertaal is het meest effectief?

Er zijn meerdere programmeertalen die geschikt kunnen zijn voor het ontwikkelen van software voor machine learning en computer vision. Bij het kiezen van de juiste taal komen vooral C++, Java en Python naar voren. (Cass, 2024)

C++ wordt vaak gebruikt wanneer snelheid en efficiëntie het belangrijkst zijn, bijvoorbeeld bij real-time toepassingen, robotica of software die op apparaten met beperkte rekenkracht draaien. Door de manier waarop C++ met geheugen en hardware is geïntegreerd, kunnen developers makkelijker hun programma beter maken zodat het beter kan presteren. Hierdoor werkt C++ dus beter als iets snel verwerkt moet worden en zo min mogelijk vertraging heeft.

Het nadeel van C++ is dat het minder minder goed is voor experimenten die snel uitgevoerd moeten worden of het ontwikkelen van nieuwe modellen. In tegenstelling tot Python heeft C++ niet dezelfde uitgebreide bibliotheken zoals TensorFlow of PyTorch, die veel van het werk voor machine learning en computer vision al doen. Dit betekent dat je meer zelf moet programmeren en debuggen, wat het ontwikkelen van nieuwe algoritmes veel langer en ingewikkelder maakt. (Trigolos, sd)

Java staat erom bekend dat het heel stabiel is en dat programma's die in Java geschreven zijn op allerlei verschillende systemen en apparaten kunnen draaien. Dus blijven programma’s in java betrouwbaar werken, ook in grote en complexe projecten. Deze eigenschappen maken Java populair voor algemene softwareontwikkeling, vooral wanneer een systeem lange tijd moet blijven draaien zonder fouten.

In machine learning en computer vision wordt Java echter veel minder gebruikt. De belangrijkste reden hiervoor is dat de meeste moderne bibliotheken en frameworks, zoals TensorFlow, PyTorch en OpenCV, vooral in Python beschikbaar zijn. Deze bibliotheken bieden kant-en-klare functies voor het bouwen, trainen en testen van modellen, waardoor het werk veel sneller en eenvoudiger gaat. In Java bestaan vergelijkbare tools wel, zoals Weka en H2O, maar deze zijn vaak minder uitgebreid en minder makkelijk te gebruiken, vooral voor beginners.

Daarnaast is de community rond Java voor machine learning kleiner dan die van Python. Dit betekent dat er minder uitleg video’s, voorbeelden en ondersteuning beschikbaar zijn. Hierdoor wordt Java meestal alleen gekozen wanneer er specifieke redenen zijn, zoals bestaande systemen die al in Java zijn geschreven, of wanneer stabiliteit en compatibiliteit boven alles gaan. (Deremuk, 2024)

Voor dit project is Python dus het meest geschikt, vooral omdat je er makkelijk mee kunt experimenteren, snel modellen kunt bouwen, trainen en testen, en sneller dan in C++ of Java fouten kunt terugvinden. Ook werkt Python goed samen met robotica-systemen zoals ROS, wat handig is als je de software op een echte robot wilt laten werken. Daarom is Python voor dit project de meest logische en overzichtelijke keuze. (Davis, 2025)

Machine learning (ML) voor visuele inspectie

Om ML voor visuele inspectie te begrijpen en te kunnen toepassen moeten we eerst een beter begrip hebben van hoe een ML model “leert”. Hierbij wordt er gedifferentieerd tussen “supervised” learning en “representational” learning.

Supervised learning (onder begeleid leren) is een zeer belangrijk onderdeel van machine learning. Eigenlijk is het de meest gebruikte methode. Het komt hierop neer: je geeft een computerprogramma een heleboel voorbeelden waarbij je al weet wat het goede antwoord is.

Stel je voor: je voedt het algoritme met duizenden gelabelde foto's. Bij elke foto zeg je: "dit is een kat" of "dit is een hond". Het algoritme gaat dan op zoek naar patronen in de data, welke pixels, vormen of kleuren horen meestal bij een kat, en welke bij een hond? Het leert de relatie tussen de foto, de invoer dus en het label, de uitvoer. Als het genoeg voorbeelden heeft gezien, kun je het een compleet nieuwe foto laten zien, en het kan dan zelf voorspellen of het een kat of een hond is.

Dit wordt niet alleen gebruikt voor plaatjes. Supervised learning wordt overal toegepast: in Natural Language Processing (NLP, zoals chatbots), beeld- en videoclassificatie, in de medische industrie bij analyses om bijvoorbeeld scans te interpreteren, en nog vele andere toepassingen.

Er zijn allerlei verschillende technieken en algoritmes voor supervised learning, elk met hun eigen voor- en nadelen. Sommige zijn sneller, andere zijn nauwkeuriger. Een groot risico is overfitting: dan leert het model je trainingsvoorbeelden zo goed dat het ze uit z'n hoofd kent, maar kan het geen nieuwe, onbekende data meer goed voorspellen. Daarom is het belangrijk om het model goed te testen met meetwaarden. Onder andere; **accuracy**, **precision** en **recall**.

Supervised learning is dus de basis voor een heleboel slimme technologie die we elke dag tegenkomen. (Tiwari, 2022)

Nu het concept van “representational” learning. Representationeel leren is een heel belangrijk onderdeel van moderne AI, vooral in deep learning. Het gaat erom dat een computermodel zelf nuttige kenmerken leert uit ruwe data, zoals foto's. In plaats van dat een programmeur handmatig regels schrijft dat bijvoorbeeld vuil een donkere vlek is, kijkt het model zelf naar vele voorbeelden en ontdekt zelf de onderliggende patronen. Deze patronen worden opgeslagen als een soort digitale vingerafdruk, een 'embedding'. Deze techniek is heel goed te gebruiken voor taken zoals het zoeken en terugvinden van vergelijkbare afbeeldingen. Door de geometrische structuur van de data te leren of feedback van gebruikers te gebruiken, kan het model steeds betere en efficiëntere voorstellingen maken, zelfs voor dingen die het tijdens de training nog nooit heeft gezien.

(Maria Tzelepi, 2022)

Hierop volgt de werking van “feature extraction” tegenover “End to end learning”.

Feature extraction is een belangrijke stap in machine learning. Het is het proces waarbij je de nuttige, karakteristieke informatie uit ruwe data haalt, zoals een afbeelding. Stel je hebt een foto van een zonnepaneel, daar zit ontzettend veel informatie in, maar niet alles is relevant om bijvoorbeeld vuil te spotten. Feature extraction zet die ruwe data om in een reeks specifieke, meetbare kenmerken, zoals textuurpatronen in een afbeelding. Voor deze stap is vaak veel wiskundige analyse nodig. Het doel is om een compacte, machine leesbare “vingerafdruk” van de data te maken die het belangrijkste opvangt, zodat het model er daarna beter en sneller mee kan werken. Het is dus niet zomaar data verwerken, maar echt de verborgen, karakteristieke informatie analyseren die nodig is voor de specifieke taak. (M. Menagadevi, 2024)

“End to end learning” is juist wat we zien bij deep learning. In plaats van dat je als programmeur eerst zelf handmatig de belangrijke kenmerken oftewel features, uit ruwe data moet halen, zoals randen of hoeken in een afbeelding en die pas daarna aan een model geeft, doet een deep learning model dit alles in één keer. Het model krijgt de ruwe data, bijvoorbeeld de ruwe pixels van een foto en leert zelf, via zijn verschillende lagen, welke features het belangrijkst zijn voor de taak. De eerste lagen leren simpele dingen zoals randen, en diepere lagen combineren die tot complexe patronen en objecten. Omdat het model precies de features leert die nuttig zijn voor het specifieke doel, zoals het herkennen van vuil, is het veel efficiënter en nauwkeuriger dan de oude methode waar je riskeerde dat verkeerde of onnodige features het model in de war brachten.

(Abtin Mahyar, 2025)

Keuze voor de moderne aanpak

Zoals hierboven beschreven, berust traditionele beeldverwerking met supervised learning dus op een twee-staps proces. Eerst worden via technieken voor feature extraction specifieke, meetbare kenmerken handmatig uit de ruwe data gehaald. Vervolgens wordt een supervised learning-model getraind met deze vooraf gedefinieerde features om voorspellingen te doen. Deze aanpak is beperkt voor taken met veel variabiliteit, zoals het detecteren van vuil op zonnepanelen. De complexiteit en onvoorspelbaarheid van vuilpatronen, in vorm, textuur en kleur, maken het praktisch onmogelijk om van tevoren de perfecte set handmatige features te definiëren die alle variaties betrouwbaar kan onderscheiden. Hierdoor loopt het model het risico irrelevante of misleidende kenmerken te gebruiken, wat slecht is voor de nauwkeurigheid.

Dit is precies de beperking die wordt opgelost door representational learning en end-to-end learning. In plaats van afhankelijk te zijn van voorgeprogrammeerde regels, leert een model via representational -learning zelf de onderliggende, betekenisvolle patronen uit de data, die het opslaat als efficiënte 'embeddings'. End-to-end learning binnen deep learning gaat nog een stap verder: Het elimineert de handmatige feature extractie gedeelte volledig. Het model verwerkt de ruwe pixels en leert geïntegreerd in één training welke features gebaseerd op een hiërarchie van simpele randen tot complexe texturen, het belangrijkst zijn voor de specifieke taak. Door deze geautomatiseerde aanpak kan het model zich aanpassen aan de enorme variatie in echte vuilpatronen, wat leidt tot nauwkeurigere detectie dan met de klassieke methode mogelijk is.

Covolutional Neural Networks

Convolutional Neural Networks, CNN's of ConvNets, zijn een speciaal type neuraal netwerk dat is ontworpen om driedimensionale data, zoals afbeeldingen, te verwerken voor taken als classificatie en objectherkenning. Ze bestaan uit drie hoofdtypen lagen: de convolutionele laag, de poolinglaag en de fully-connected (FC) laag. In tegenstelling tot oudere methoden, waarbij feature-extractie handmatig was en veel tijd kostte, zijn CNN’s een schaalbare aanpak. Ze zijn superieur in prestaties bij het verwerken van beeld-, spraak- of audiogegevens. Hun kracht ligt in het hiërarchisch leren van features, daarom zijn ze zo geschikt voor afbeeldingen.

De kern van een CNN is de convolutionele laag. Hierbij schuift een kleine filter of kernel over de invoerafbeelding. Op elke positie berekent hij een dot product tussen de filterwaarden en de onderliggende pixels, wat resulteert in één getal in een feature map wat ook wel activatiemap wordt genoemd. Dezelfde filter, met dezelfde gewichten wordt over de hele afbeelding gebruikt, wat parameter sharing heet. De eerste lagen van een CNN halen de eenvoudige, laag-niveau features uit zoals kleuren en randen. Hoe verder de data door de lagen van het netwerk gaat, hoe grotere en complexere vormen het

herkent. Uiteindelijk kan het netwerk zo het juiste object herkennen. Dit wordt hiërarchisch feature learning genoemd.
Zoals verwacht is het trainen van een diep CNN van scratch iets wat enorme hoeveelheden gelabelde data en aanzienlijke rekenkracht vereist, vaak met speciale hardware zoals GPU's. Dit is vaak onpraktisch. Daarom zijn pretrained CNN backbones zo gebruikelijk. Dit zijn CNN's, zoals de bekende LeNet-5, die al zijn voorgetraind op gigantische, algemene datasets zoals ImageNet. Tijdens dit voortrainen hebben ze al een rijke bibliotheek van algemene visuele features geleerd, van randen tot complexe texturen. Deze getrainde CNN-backbone kan vervolgens als een krachtig startpunt worden genomen voor een nieuwe, specifiekere taak, zoals vuil detecteren op zonnepanelen. Alleen de laatste lagen hoeven dan te worden aangepast en getraind met een veel kleinere, eigen dataset. Dit maakt "state-of-the-art" computervisie efficiënt en toegankelijk voor de meeste mensen.

(What are convolutional neural networks? , 2025)

Siamese Neural Networks

Een Siamese Neural Network bestaat uit twee of meer precies dezelfde subnetwerken die samen gewichten delen. Het netwerk is speciaal ontworpen voor vergelijking. In plaats van dat het één afbeelding direct in een categorie indeelt, zoals een gewone classificator doet, verwerkt het twee afbeeldingen tegelijk. Beide afbeeldingen gaan door identieke netwerken, die elk een uitvoervector, ook wel een embedding genoemd, maken. Vervolgens worden deze twee vectoren met elkaar vergeleken om te bepalen hoe sterk ze op elkaar lijken.

Dit heet metric learning of metriekleren. Het netwerk leert niet om namen van categorieën te voorspellen, maar het leert een slimme manier om de onderlinge afstand tussen gegevens te meten. Het doel is om de afbeeldingen zo om te zetten in getallen (embeddings) dat soortgelijke dingen ook op de getallenlijn dicht bij elkaar komen te liggen. Dit gebied waar die getallen in liggen wordt de latente ruimte genoemd. Tijdens de training wordt het netwerk getraind om de afstand tussen bij elkaar horende afbeeldingen klein te maken en de afstand tussen niet-bij elkaar horende afbeeldingen groot.

Dit principe is perfect voor het vinden van vuil op zonnepanelen. Je kunt het netwerk trainen met voorbeelden van schone panelen. Het leert dan hoe een schoon paneel er in getallen uit ziet, en het slaat dat patroon op als een referentie in de latente ruimte. Later, als je een nieuw paneel controleert, maakt het netwerk een embedding van die nieuwe foto. Vervolgens meet het hoe ver deze nieuwe getallenreeks af ligt van de bekende, schone referentie. Vuil wordt gedetecteerd als een afwijking van die schone referentie. Een grote afstand in de latente ruimte betekent dat het nieuwe paneel afwijkt van wat schoon is, en dus waarschijnlijk vuil bevat. Dit werkt zelfs als je maar een paar voorbeelden van schone panelen hebt om mee te trainen. (Siamese Neural Network , 2021)

Image Embeddings en Afstandsgebaseerde Detectie

Een embedding vector is een vector met een vaste lengte die een numerieke representatie vormt van de inhoud van een afbeelding. Het is een vorm van dimensionality reduction waarbij hoge-dimensionale pixeldata wordt geprojecteerd naar een lager-dimensionale ruimte die de meest kenmerkende visuele eigenschappen behoudt. Deze vectoren worden gegenereerd door modellen zoals convolutional neural networks (CNN's), die kenmerken hiërarchisch eruit halen en coderen als punten in een embedding space. (Nwoke, 2023)

In deze ruimte kan de wiskundige afstand tussen twee vectoren geïnterpreteerd worden als hun visuele gelijkenis: vergelijkbare afbeeldingen liggen dicht bij elkaar, terwijl ongelijke objecten ver uit elkaar liggen. (Embeddings: Embedding space and static embeddings, sd)

De gelijkenis tussen de twee embedding vectoren wordt meestal gemeten met Euclidean distance en cosine similarity. De Euclidean distance meet de directe 'rechte lijn' afstand tussen twee punten in de vectorruimte. Vectoren met een kleine Euclidean distance bevinden zich in hetzelfde gebied van de ruimte en hebben een vergelijkbare soort kenmerken. Cosine similarity meet daarentegen de hoek tussen twee vectoren en is daarom ongevoelig voor hun absolute grootte. Het geeft aan of de vectoren in dezelfde algemene richting vanuit de oorsprong wijzen, wat betekent dat ze dezelfde verhouding of patroon van kenmerken hebben, zelfs als de ene veel sterker is uitgedrukt dan de andere. De keuze tussen deze twee metingen hangt af van de taak. (Luca, 2024)

In plaats van een model direct te trainen om klasselabels zoals schoon of vuil te voorspellen, kan een aanpak gebaseerd op gelijkenis worden gebruikt met afstandsdrempels. Hierbij wordt eerst een referentie-embedding bepaald, bijvoorbeeld van een schoon zonnepaneel. Voor nieuwe afbeeldingen wordt de afstand tot deze referentie berekend. Blijft deze afstand onder een vooraf ingestelde drempelwaarde, dan wordt de afbeelding als vergelijkbaar met schoon beschouwd. Overschrijdt de afstand de drempel, dan wordt de afbeelding geclassificeerd als afwijkend of vuil. De keuze van de drempel bepaalt de balans tussen precisie en recall: een hogere drempel vermindert fout-positieven, terwijl een lagere drempel zorgt voor hogere gevoeligheid en minder gemiste vervuiling.
(How to use classification threshold to balance precision and recall, 2025)

Data Preprocessing en Normalisatie

In het domein van machine learning, en specifiek bij neurale netwerken, is het een belangrijk principe om alle inputdata te voorzien van een uniforme schaal. Dit proces heet normalisatie. Het doel is om te voorkomen dat kenmerken met grotere numerieke waarden onevenredig veel invloed krijgen op het leerproces van het model.

Een simpele en veel gebruikte manier om dit te doen is door de waarden om te schalen naar een vast, klein bereik, zoals tussen 0 en 1. Deze vorm van normalisatie, min-max normalisatie genoemd, zorgt ervoor dat alle waarden proportioneel binnen dit vaste kader vallen. Voor afbeeldingsdata betekent dit in de praktijk dat de ruwe pixelwaarden, die bijvoorbeeld kunnen liggen tussen 0 en 255, worden gedeeld door de maximale waarde (255) om ze om te zetten naar het bereik [0, 1].

De reden om deze normalisatie toe te passen heeft te maken met de manier waarop een neuraal netwerk leert. Tijdens het trainen wordt gebruikgemaakt van wiskundige optimalisatie, en dit proces is gevoelig voor de schaal van de invoer. Wanneer de invoerwaarden op een consistente en genormaliseerde schaal staan, kan het netwerk efficiënter en stabieler leren. Zonder deze consistentie zouden heldere pixels met een hoge waarde te veel invloed kunnen krijgen ten opzichte van donkere pixels, wat het leerproces minder betrouwbaar en minder efficiënt maakt. (Jaiswal, 2024)

Tile-Based Image Analysis voor Geavanceerde Inspectie

Voor de analyse van hoge-resolutiebeelden, zoals opnames van zonnepanelen, is een score voor het hele beeld onvoldoende, omdat het geen informatie geeft over waar een eventueel probleem zich bevindt. Om deze spatiale lokalisatie te bereiken zonder dure pixel-per-pixel segmentatie, wordt tile-based of patch-based analyse toegepast. Hierbij wordt het grote beeld opgedeeld in een raster van kleinere vierkantjes oftewel tiles. (Rajshekhar, 2022)

Deze aanpak maakt gebruik van een sliding window dat over de afbeelding schuift. Om te voorkomen dat objecten of afwijkingen precies op de randen van tiles vallen en worden gemist, worden deze windows vaak overlappend gemaakt, wat de dekking en detectienauwkeurigheid verbetert.

Deze methode sluit aan bij het concept van weakly supervised learning. Hierbij krijgt een afbeelding alleen een algemeen label, bijvoorbeeld vies, maar hoeft niet voor elke pixel aangegeven te worden waar het vuil precies zit, wat volledige segmentatie zou vereisen. Het model leert zelf om binnen de afbeelding de specifieke tiles te identificeren die het sterkst bijdragen aan dat label. (Dogra, 2020)

Voor elke tile kan een model zoals een Convolutional Neural Network (CNN) of een Siamese Network een embedding genereren, een vaste lengte vector die de visuele

kenmerken samenvat. Door de embedding van een tile te vergelijken met die van een referentiebeeld van een schoon paneel, bijvoorbeeld met een afstandsmaat, ontstaat er een anomaliescore per tile.

Deze plaatsgebonden scores vormen de basis voor een heatmap. Een heatmap is een handige vorm van data visualisatie die numerieke waarden omzet in een overzichtelijk kleurenspectrum, bijvoorbeeld van blauw, laag risico/gelijk aan de referentie, naar rood wat hoog risico/afwijkend voorstelt. Dit maakt de spatiale verdeling (spatial averaging) van het probleem meteen duidelijk.

(What is Heatmap Data Visualization and How to Use It?, 2024)

Door de gekleurde tiles terug te leggen op hun oorspronkelijke positie, ontstaat er een visueel overzicht dat in één oogopslag de locatie van potentiële vuilplekken laat zien. Deze hele aanpak van het opdelen in tiles tot het genereren van een heatmap is precies hoe geavanceerde beeldanalyse uitgevoerd kan worden ondanks de beperkte, zwakke labels.

Dit is verder ook nog van groot belang voor debugging. Een goede heatmap laat niet zomaar willekeurige gekleurde vlekken zien, maar duidelijke, samenhangende patronen die logisch zijn. Als de heatmap er rommelig uitziet of onlogische patronen laat zien, bijvoorbeeld sterk afwijkende scores verspreid over een verder schoon paneel, is dat een duidelijk signaal dat er iets mis kan zijn in het model, de data of de gekozen drempelwaarden. De heatmap werkt dus als een visueel controle-instrument dat helpt de betrouwbaarheid van het hele inspectiesysteem te beoordelen. (Yi, 2025)

Computer Vision met OpenCV

OpenCV (Open Source Computer Vision Library) is een open-source library voor computer vision en beeldverwerking, met een uitgebreide verzameling algoritmen voor het inlezen, verwerken en analyseren van beelden en video. (Alvi, 2023)

Het proces begint met “image acquisition” (afbeeldingen verkrijgen). OpenCV kan eenvoudig afbeeldingen van een bestand inlezen, maar ook direct frames verkrijgen van een aangesloten camera, wat essentieel is voor real-time inspectiesystemen. Als het eenmaal verkregen is, begint de voorbewerking (preprocessing). Dit zijn de belangrijkste stappen om de ruwe beelddata voor te bereiden voor latere, complexere analyse. Veelvoorkomende bewerkingen zijn het omzetten van kleurenafbeeldingen naar grijswaarden (om complexiteit te verminderen) en het toepassen van filters, zoals Gaussian blur, om ruis te onderdrukken en de beeldkwaliteit te verbeteren. (Alvi, 2023)

Om specifieke objecten zoals zonnepanelen te isoleren, worden technieken voor maskering en randdetectie gebruikt. Maskering kan gebaseerd zijn op kleur of

helderheid, waarbij pixels die binnen een bepaalde drempelwaarde vallen worden geselecteerd om een binaire maskerafbeelding te creëren.

Voor het vinden van de exacte contouren van een object is randedetectie vaak effectiever. De “Canny edge” detector is een veelgebruikte methode hiervoor binnen OpenCV. Deze detector vindt randen door eerst ruis te verminderen, vervolgens de gradiënt dus de sterkte van helderheidsverandering te berekenen, en tenslotte met een slim algoritme alleen de sterkste en meest samenhangende randen over te houden. Het resultaat is een duidelijk beeld van de randen en contouren van objecten in het plaatje, wat de eerste stap is om een individueel paneel te identificeren en te isoleren voor verdere analyse. (Team, 2021)

Path Planning/ Mapping

Path planning is een fundamenteel probleem in de robotica. Het gaat om het vinden van een pad zonder botsingen voor een robot vanaf zijn startpositie naar een doelpositie. Dit heet ook wel het plannen van een collision-free path. Het ideale, complete algoritme vindt altijd een pad als het bestaat en stopt ook op tijd als dat niet zo is. Omdat dit probleem erg ingewikkeld is voor complexe robots, zijn moderne algoritmen vooral praktisch: ze moeten een goede balans vinden tussen betrouwbaarheid en efficiëntie. Enkele veelgebruikte methoden zijn Artificial Potential Fields, waarbij de robot wordt aangetrokken door het doel en afgestoten door obstakels, en Rapidly-exploring Random Trees (RRT), een slimme methode die door willekeurige steekproeven een boomstructuur groeit van start naar doel.

(Frank Dellaert, 2023)

Ultrasone Sensoren

Ultrasone sensoren zijn instrumenten die de afstand tot een object meten door gebruik te maken van ultrasone geluidsgolven. Dit zijn geluidsgolven met een frequentie boven de 20 kHz, die het menselijk oor niet kan horen. De sensor werkt volgens het time-of-flight principe: een zender stuurt een ultrasone puls uit. Als deze puls een object raakt, kaatst deze terug naar de sensor waar een ontvanger hem opvangt. De sensor meet de tijd tussen het uitzenden en ontvangen. Omdat de geluidssnelheid in lucht bekend is (ongeveer 340 m/s), kan de afstand tot het object worden berekend met de formule: afstand = (geluidssnelheid * gemeten tijd)/ 2. Een groot voordeel van deze techniek is dat deze onafhankelijk werkt van lichtomstandigheden, rook, stof of de kleur van het object. Hierdoor is de sensor bijvoorbeeld geschikt voor het detecteren van transparante objecten waar optische sensoren zouden falen. Ze worden daarom vaak ingezet voor taken zoals aanwezigheidsdetectie, afstandsmeting en het voorkomen van botsingen bij robots. Een voorwaarde is wel dat het object het geluid reflecteert.

(Understanding Ultrasonic Sensor, 2023)

Raspberry pi

Een Raspberry Pi is een compacte en betaalbare single board computer. In tegenstelling tot een microcontroller, zoals een Arduino, is een Raspberry Pi een hele computer die een besturingssysteem kan draaien. Dit betekent dat hij in staat is om meerdere programma's tegelijk uit te voeren, bestanden op te slaan en complexe berekeningen uit te voeren.

Raspberry pi's worden vaak gebruikt in toepassingen waar een normale computer te groot, te duur of te energie intensief is. Door het kleine formaat en het lage stroomverbruik zijn ze geschikt voor gebruik in mobiele systemen, zoals robots, slimme apparaten en meetopstellingen. Tegelijkertijd hebben ze voldoende rekenkracht om beeldverwerking, netwerkcommunicatie en zelfs eenvoudige machine learning taken uit te voeren.

Een belangrijke kenmerk van de raspberry pi is dat hij een algemene invoer en uitvoerpinnen (GPIO) heb. Hiermee kunnen sensoren, motoren en andere elektronische componenten direct worden aangesloten.

Binnen robotica word raspberry pi erg vaak ingezet als centrale besturingseenheid. Hij verwerkt sensorgegevens, neemt beslissingen op basis van programma's en stuurt motoren aan.

Hoewel de raspberry pi niet zo krachtig is als een laptop of desktop computer, kan hij toch effectief worden gebruikt voor machine learning toepassingen. Dit gebeurt meestal door het model niet op de raspberry pi zelf de trainen. Het trainen van een machine learning model heeft veel rekenkracht en tijd nodig, wat beter geschikt is voor een laptop of server.

(about, n.d.)

Unity Engine

Unity is een veelgebruikte software engine die oorspronkelijk is ontwikkeld voor het maken van videogammes, maar tegenwoordig ook wordt toegepast voor simulatie en technische visualisatie. Met Unity kunnen driedimensionale omgevingen worden opgebouwd waarin objecten zich realistisch gedragen door middel van een ingebouwde physics engine. Programmmeren in Unity gebeurt met de programmeertaal C#. (Physics, n.d.)

Unity wordt vaak gebruikt in situaties waarin systemen eerst virtueel getest moeten worden voordat ze in de echte wereld worden toegepast. Door objecten te voorzien van massa, botsingen en beweging kan het gedrag van machines en robots worden nagebootst zonder risico op schade.

Binnen robotica wordt Unity gebruikt om robots en hun omgeving te simuleren. Sensoren zoals camera's en afstandssensoren kunnen virtueel worden gesimuleert, waardoor algoritmes voor navigatie en beeldverwerking getest kunnen worden. Dit maakt het

mogelijk om software te ontwikkelen en te testen voordat deze op echte hardware wordt ingezet. (Unity, n.d.)

Hoewel Unity zelf geen besturingseenheid is, vormt het een belangrijk hulpmiddel bij het ontwerpen en testen van robotsystemen. De resultaten uit simulaties kunnen later worden toegepast op fysieke robots, zoals robots die worden aangestuurd door een Raspberry Pi.

Concept

Op basis van de theoretische achtergrond hebben we het concept ontwikkeld, dat we hieronder toelichten.
Het schoonmaakproces is opgesplitst in drie fasen:

Fase 1 Mappen van het paneel

In deze fase tekent het systeem het zonnepaneel als een kaart met behulp van verschillende sensoren. Op basis van deze data wordt een virtuele kaart opgebouwd, waarbij een coördinatensysteem ontstaat. Binnen deze virtuele kaart wordt een schoonmaak pad bepaald, dat het systeem vervolgens zal volgen.

Fase 2 Live schoonmaak en detectie

Tijdens het schoonmaken staat de camera continu achter de robot. De live beelden worden doorgestuurd naar een centrale computer, waar machine learning wordt toegepast om viezigheid te detecteren en de coördinaten ervan te noteren.

Fase 3 Nacontrole en herbehandeling

Na het voltooien van het standaard schoonmaak pad keert het systeem terug naar de coördinaten die tijdens fase 2 als vies zijn gemarkeerd, om deze plekken opnieuw schoon te maken.

Mogelijke methoden

Om de drie fasen uit te voeren, hebben we verschillende methodes bedacht:

Manier 1

Machine learning wordt gebruikt om het verschil tussen schoon en vies te leren op een laptop. Het getrainde model wordt vervolgens op het systeem zelf geïmplementeerd, zodat de robot kan bepalen welke delen van het paneel schoon zijn.

Manier 2

Aan het begin wordt een deel van het paneel meerdere keren schoongemaakt om zeker te zijn dat het 100% schoon is. Daarna wordt een foto gemaakt, die gebruikt wordt om live beelden van de camera te vergelijken op kleurverschillen om viezigheid te detecteren. Dit is meer klassieke computer vision dan machine learning.

Manier 3

Net als bij manier 2 wordt een deel van het paneel meerdere keren schoongemaakt tot het volledig schoon is. Vervolgens wordt er een machine learning model getraind op deze referentiefoto om verschillen te detecteren in de live camera beeld, zodat de coördinaten van vuil worden genoteerd.

Schoonmaak proces

Initiatie

De robot wordt geplaatst in de linkeronderhoek van het zonnepaneel. Vervolgens wordt het programma gestart.
Na het opstarten activeert de robot de ultrasone sensoren en bereidt het systeem zich voor op het mappen van het zonnepaneel.

Mappen van het zonnepaneel

Tijdens de map fase volgt de robot de rand van het zonnepaneel met behulp van ultrasone sensoren.
De robot heeft twee ultrasone sensoren: een aan de voorkant en een aan de linkerzijde.

De voorste sensor detecteert of er nog een oppervlak voor de robot aanwezig is. Wanneer deze sensor geen object meer meet, betekent dit dat de robot een rand van het paneel heeft bereikt.
Tegelijkertijd controleert de linker sensor of er nog steeds een rand naast de robot aanwezig is. Wanneer dit het geval is, herkent het systeem dit als een hoekpunt.

Bij het detecteren van een hoek draait de robot 90 graden naar rechts en vervolgt hij het volgen van de rand. Dit proces wordt herhaald totdat de robot weer terugkeert naar het startpunt. Op deze manier wordt de volledige omtrek van het zonnepaneel in kaart gebracht. Hiermee wordt een navigatie gepland waar de robot zal gaan volgen.

Navigatie volgen

De robot volgt tijdens het schoonmaken een spiraalvormig pad naar binnen, zodat het volledige oppervlak van het zonnepaneel systematisch wordt gereinigd.
Tijdens het volgen van dit pad is de camera aan de achterzijde van de robot continu actief.

De camerabeelden worden gebruikt om vervuiling te detecteren. Wanneer vuil wordt herkend, slaat het systeem de bijbehorende coördinaten op. Deze locaties worden niet direct opnieuw gereinigd, maar gemarkeerd zodat de robot hier in een latere fase naar kan terugkeren voor extra schoonmaak.

Terugkeren naar vervuilde gebieden

Na het voltooien van het reguliere schoonmaak-pad worden alle opgeslagen coördinaten van vervuilde gebieden geanalyseerd. Op basis van deze gegevens wordt een nieuw navigatie-pad berekend.

De robot volgt dit pad om gericht terug te keren naar de eerder gemarkeerde locaties en deze opnieuw schoon te maken. Op deze manier krijgen hardnekkige vervuilingen extra aandacht, zonder dat het volledige zonnepaneel opnieuw gereinigd hoeft te worden.

Uitwerking Onderzoek

Vanuit het concept is onderzocht welke stappen noodzakelijk zijn om het systeem in de praktijk te realiseren. Daarbij lag de focus op het toepassen van machine learning voor vuildetectie en op het bepalen van een optimale werkwijze vanuit het perspectief van computersystemen.

Het onderzoek is opgebouwd rond de volgende deelvragen:

1. Op welke manier kan machine learning worden toegepast voor het detecteren van vervuiling op zonnepanelen?
2. Wat is efficiënt vanuit het perspectief van computers?
3. Hoe kan een efficiënte navigatie-pad worden bepaald tussen gedetecteerde vervuilde gebieden op een zonnepaneel?
4. Hoe kan een machine learning-model worden getraind en toegepast om vervuiling op zonnepanelen betrouwbaar te detecteren?

Op welke manier kan machine learning worden toegepast voor het detecteren van vervuiling op zonnepanelen?

Hypothese

Machine learning kan worden gebruikt om vervuiling op zonnepanelen te detecteren door camerabeelden te vergelijken met een schoon referentiebeeld. Een Siamese Neural Network is hiervoor geschikt, omdat dit netwerk kan bepalen hoe sterk twee afbeeldingen op elkaar lijken.

Methode en materiaal

Voor dit experiment is gebruikgemaakt van een Siamese Neural Network. Dit netwerk vergelijkt twee afbeeldingen en berekent hoe groot het verschil tussen deze afbeeldingen is.

Hier zijn de exacte stappen die we hebben genomen voor onze methode:

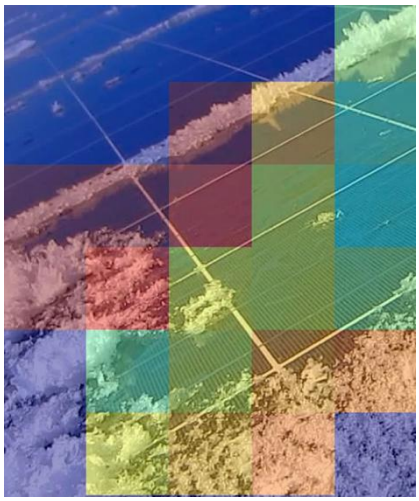
1. Er is eerst een foto gemaakt van een volledig schoon zonnepaneel.
2. Vervolgens zijn meerdere foto's gemaakt van zonnepanelen met verschillende soorten vervuiling, zoals stof en vlekken.
3. Alle foto's zijn op een laptop geladen en omgezet naar hetzelfde formaat en dezelfde helderheid.
4. Het Siamese Neural Network is gebruikt om van elke foto een numerieke representatie (embedding) te maken.
5. De afstand tussen de embedding van het nieuwe beeld en de embedding van het schone referentiebeeld is berekend.
6. Als deze afstand groter was dan een ingestelde drempelwaarde, werd het gebied als vervuild beschouwd.

Materialen:

1. Digitale camera
2. Laptop met Python
3. Machine learning-bibliotheken (TensorFlow, Keras via Tensorflow, Numpy, OpenCV)
4. Dataset bestaande uit afbeeldingen van schone en vervuilde zonnepanelen

Resultaten

Het model herkende duidelijke vervuiling goed. Wanneer het zonnepaneel zichtbaar vuil was, was het verschil met het schone referentiebeeld groot. Bij lichte vervuiling of sterke reflectie werd soms onterecht vuil gedetecteerd.



Conclusie

De hypothese wordt bevestigd. Met behulp van een Siamese Neural Network kan vervuiling op zonnepanelen worden gedetecteerd door afwijkingen ten opzichte van een schoon referentiebeeld te meten.

Wat is efficiënt vanuit het perspectief van computersystemen?

Efficiëntie in computersystemen verwijst naar het uitvoeren van berekeningen met zo min mogelijk gebruik van rekenkracht, geheugen en tijd, terwijl de gewenste output behouden blijft. Een efficiënt algoritme vermijdt onnodige berekeningen en verwerkt data in een zo compact mogelijke vorm. Dit is vooral belangrijk bij toepassingen waarin grote hoeveelheden data herhaaldelijk worden verwerkt, zoals bij beeldanalyse en automatische inspectiesystemen. In zulke toepassingen bepaalt de efficiëntie in grote mate de snelheid, schaalbaarheid en praktische inzetbaarheid van het systeem. (Spasojevic, 2024)

Hypothese

Voor het detecteren van vuil op zonnepanelen is een methode gebaseerd op het vergelijken van image embeddings efficiënter dan directe pixel-per-pixel vergelijking. Deze efficiëntie uit zich in een kortere verwerkingstijd en een lagere rekencomplexiteit, doordat embeddings afbeeldingen samenvatten in compacte vectoren die alleen de belangrijkste visuele kenmerken bevatten.

Methode en Materiaal

Om de hypothese te toetsen zijn twee methoden met elkaar vergeleken: directe pixel-per-pixel vergelijking en vergelijking op basis van embedding vectoren. Beide methoden zijn toegepast op dezelfde set digitale RGB-afbeeldingen.

Bij de embedding-gebaseerde methode is een vooraf getraind convolutional neural network (CNN) gebruikt als feature extractor. Voor elke afbeelding werd één embedding vector gegenereerd met een vaste lengte van 128 dimensies. Deze vector representeert de belangrijkste kenmerken van de afbeelding. Afbeeldingen zijn vervolgens met elkaar vergeleken door de afstand tussen hun embedding vectoren te berekenen met behulp van Euclidische afstand en cosine similarity.

Ter vergelijking is een pixel-per-pixel methode toegepast waarbij alle RGB-pixelwaarden van twee afbeeldingen direct met elkaar worden vergeleken. Bij afbeeldingen van 256 × 256 pixels komt dit neer op 196.608 afzonderlijke waarden per afbeelding.

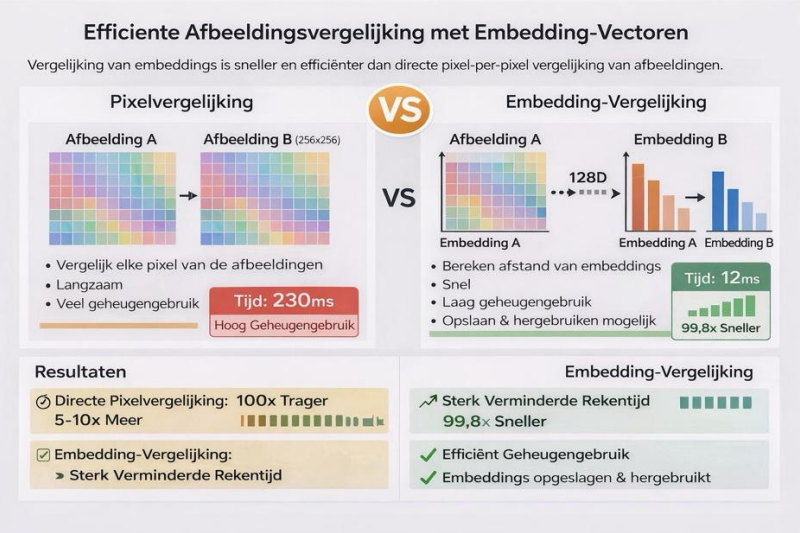
De efficiëntie van beide methoden is beoordeeld op basis van rekentijd, hoeveelheid verwerkte data en de mogelijkheid tot hergebruik van berekende resultaten.

Materiaal:
Voor dit onderzoek zijn digitale RGB-afbeeldingen gebruikt en is een vooraf getraind convolutional neural network toegepast om embedding vectoren te genereren. De vergelijkingen zijn uitgevoerd in een Python-omgeving.

Resultaten

Uit de resultaten blijkt dat de embedding-gebaseerde methode veel efficiënter is dan directe pixel-per-pixel vergelijking. Bij de pixelvergelijking moesten alle individuele pixelwaarden worden verwerkt, wat resulteerde in een hoge rekentijd en een groot geheugengebruik.

De embedding-gebaseerde methode verminderde de hoeveelheid verwerkte data van ongeveer 200.000 waarden per afbeelding naar slechts 128 waarden. Hierdoor nam de rekentijd sterk af en was het geheugengebruik aanzienlijk lager. Daarnaast konden embedding vectoren worden opgeslagen en hergebruikt, waardoor bij meerdere vergelijkingen geen herhaalde berekeningen door het CNN nodig waren. Hoewel de exacte tijden kunnen variëren per systeem, blijft het relatieve verschil in rekencomplexiteit tussen beide methoden constant.



Figuur 12 Schematisch visualisatie van het uitgevoerde experiment

De bovenstaande figuur toont een vergelijking tussen directe pixel-per-pixel afbeeldingsvergelijking en embedding-gebaseerde afbeeldingsvergelijking. Hierbij worden de verschillen in datavolume, rekentijd en geheugengebruik inzichtelijk gemaakt.

Conclusie

Op basis van de resultaten kan worden geconcludeerd dat het vergelijken van afbeeldingen met behulp van embedding vectoren efficiënter is dan directe pixel-per-pixel vergelijking. De hypothese wordt hiermee bevestigd.

Doordat embeddings een compacte representatie vormen van de belangrijkste visuele kenmerken van een afbeelding, zijn minder berekeningen nodig en wordt minder geheugen gebruikt. Bovendien maakt het opslaan en hergebruiken van embeddings deze methode zeer geschikt voor toepassingen zoals de automatische detectie van vuil op zonnepanelen, waarbij veel vergelijkingen nodig zijn.

Hoe kan een efficiënt navigatiepad worden bepaald tussen gedetecteerde vervuilde gebieden op een zonnepaneel?

Hypothese

Het gebruik van een geschikt algoritme maakt het mogelijk om een efficiënt navigatiepad te bepalen tussen meerdere vervuilde gebieden op een zonnepaneel.

Methode en materiaal

Tijdens fase 2 van het schoonmaakproces detecteert het systeem meerdere vervuilde gebieden en slaat het de bijbehorende coördinaten op. Deze coördinaten vormen samen een probleem waarbij de robot alle punten moet bezoeken met een zo kort mogelijk pad. Aan het begin van dit onderzoek was nog niet bekend welk algoritme hiervoor het meest geschikt zou zijn.

Om dit te bepalen is eerst onderzocht welke soorten algoritmes bestaan voor het plannen van routes langs meerdere punten. Hierbij kwam het Traveling Salesman Problem (TSP) naar voren. Dit probleem beschrijft de situatie waarin één “reiziger” alle steden precies één keer moet bezoeken met een minimale totale afstand. De situatie van de robot die meerdere vervuilde locaties moet bezoeken lijkt sterk op dit probleem. (Robinson, 1949)

Vervolgens is onderzocht hoe dit probleem kan worden opgelost. Er bestaan algoritmes die altijd het optimale pad probeert te vinden. Deze algoritmes proberen alle mogelijke volgordes van punten en kiezen daaruit de beste. Hoewel dit in theorie het kortste pad oplevert, groeit het aantal mogelijke routes extreem snel wanneer het aantal punten toeneemt. Dit betekent dat deze algoritmes veel rekentijd en geheugen nodig hebben. Voor een systeem dat draait op een Raspberry Pi is dit niet praktisch uitvoerbaar.

Daarna zijn verschillende algoritmes onderzocht. Deze algoritmes garanderen niet altijd het optimale pad, maar leveren wel snel een goed resultaat. Een bekende categorie hiervan zijn heuristieken. Heuristieken maken gebruik van eenvoudige beslisregels in plaats van het doorrekenen van alle mogelijkheden. (Allen Newell, 1976)

Een van deze heuristieken is de *greedy nearest neighbor-methode*. Bij deze methode start de robot op zijn huidige positie en kiest hij telkens het dichtstbijzijnde nog niet bezochte punt als volgende bestemming. Dit proces wordt herhaald totdat alle vervuilde gebieden zijn bezocht. Deze methode is eenvoudig te begrijpen, snel uit te voeren en vereist weinig rekenkracht. (Mhd Furqan, 2021)

Andere heuristieken, zoals simulated annealing, zijn ook onderzocht. Deze methodes kunnen soms betere paden vinden dan greedy nearest neighbor, maar vereisen meerdere iteraties, extra parameters en meer rekenkracht. Hierdoor zijn ze minder geschikt voor real-time toepassing op eenvoudige hardware. (Simulated Annealing , n.d.)

Op basis van deze vergelijking is gekozen voor de greedy nearest neighbor-methode, omdat deze methode een goede balans biedt tussen efficiëntie, eenvoud en uitvoerbaarheid.

Voor het testen van deze methode zijn de opgeslagen coördinaten van vervuilde gebieden ingevoerd in een C++ programma. Dit programma berekent de afstanden tussen de punten en bepaalt op basis daarvan de volgorde waarin de robot de locaties bezoekt.

Resultaten

De berekende routes waren logisch en overzichtelijk. De robot bezocht steeds eerst de dichtstbijzijnde vervuilde locatie, waardoor grote omwegen werden vermeden. De berekening van het pad kostte nauwelijks tijd en kon zonder problemen worden uitgevoerd op beperkte hardware.

Conclusie

De hypothese wordt bevestigd. Door gebruik te maken van een geschikt algoritme kan een efficiënt navigatiepad worden bepaald tussen meerdere vervuilde gebieden. Uit het onderzoek blijkt dat exacte oplossingen voor het Traveling Salesman Problem niet geschikt zijn vanwege hun hoge rekeneisen. De greedy nearest neighbor-methode biedt een eenvoudige en efficiënte oplossing die goed toepasbaar is binnen de beperkingen van het systeem. Daarnaast is de greedy nearest neighbor-methode geschikt voor real-time toepassing op een autonome robot, omdat het algoritme weinig rekenkracht vereist en snel beslissingen kan nemen. De vervuilde gebieden worden eerst visueel gedetecteerd met behulp van machine learning, waarna het algoritme het meest efficiënte reinigingspad bepaalt tussen deze locaties. Hoewel de greedy methode niet altijd het optimale pad oplevert, is de benadering voldoende nauwkeurig binnen de context van het schoonmaken van zonnepanelen, waar snelheid en eenvoud belangrijker zijn dan een exacte oplossing. De gekozen aanpak schaalt goed wanneer het aantal vervuilde gebieden toeneemt, in tegenstelling tot exacte oplossingen die exponentieel complexer worden. In toekomstig onderzoek kan worden gekeken naar verbeterde heuristieken om de efficiëntie verder te verhogen bij complexere scenario’s.

Hoe kan een machine learning-model worden getraind en toegepast om vervuiling betrouwbaar te detecteren?

Hypothese

Een machine learning-model dat is getraind op een laptop kan betrouwbaar worden gebruikt op een Raspberry Pi door het modelbestand te exporteren en lokaal toe te passen.

Methode en materiaal

Training:

Voor dit onderzoek is een Siamese Neural Network getraind om visuele verschillen tussen schone en vervuilde zonnepanelen te leren herkennen. In plaats van het model direct te laten voorspellen of een afbeelding schoon of vuil is, leert het netwerk om afbeeldingen met elkaar te vergelijken.

De trainingsdata bestond uit afbeeldingen van schone zonnepanelen en afbeeldingen van zonnepanelen met verschillende vormen van vervuiling, zoals stof en vogelpoep. Alle afbeeldingen zijn vooraf geschaald naar hetzelfde formaat (224×224 pixels) en genormaliseerd, zodat elke afbeelding op dezelfde manier door het netwerk wordt verwerkt.

Het model maakt gebruik van een encoder, gebaseerd op een vooraf getraind convolutional neural network. Deze encoder zet een afbeelding om in een vaste vector van 256 getallen, ook wel een embedding genoemd. Zo'n embedding bevat geen pixelinformatie meer, maar een samenvatting van de belangrijkste visuele kenmerken van de afbeelding.

Tijdens de training werden steeds twee afbeeldingen tegelijk vergeleken. Soms waren dit twee schone afbeeldingen, en soms een schone en een vervuilde afbeelding. Het model werd getraind om embeddings van vergelijkbare afbeeldingen dichter bij elkaar te brengen, en embeddings van verschillende afbeeldingen verder uit elkaar te plaatsen. Op deze manier leert het netwerk wat een schoon paneel kenmerkt en hoe vervuiling daarvan afwijkt.

Na meerdere trainingsrondes was de encoder in staat om consistente embeddings te genereren voor schone zonnepanelen. Deze encoder is vervolgens opgeslagen en vormt de basis van het detectiesysteem.

Na de training is een aparte kalibratiestap uitgevoerd. Meerdere afbeeldingen van schone zonnepanelen zijn door de encoder gehaald, waarna de resulterende embeddings zijn samengevoegd tot één referentieprofiel van een schoon paneel. Dit profiel wordt later gebruikt als vast vergelijkingspunt bij nieuwe camerabeelden.

Toepassing:

Voor de toepassing van het model is een Raspberry Pi gebruikt met een aangesloten camera. Op de Raspberry Pi draait een Python programma dat verantwoordelijk is voor het analyseren van camerabeelden.

Wanneer het programma start, worden eerst alle bestanden ingeladen. Dit zijn het getrainde machine learning model, de referentie embedding van een schoon zonnepaneel en een kalibratiebestand met grenswaarden. Vervolgens wordt de camera geactiveerd en wordt één beeld vastgelegd.

Het vastgelegde camerabeeld wordt eerst voorbereid voor analyse. Dit betekent dat het beeld wordt geschaald naar een vast formaat en dat de pixelwaarden worden genormaliseerd. Hierdoor krijgt het model altijd invoer in dezelfde vorm, wat belangrijk is voor een betrouwbare werking.

Daarna wordt het voorbereide beeld aangeboden aan het machine learning model. Het model zet dit beeld om in een embedding, vergelijkbaar met de embeddings die tijdens de training zijn gebruikt. Vervolgens wordt de afstand berekend tussen deze nieuwe embedding en de opgeslagen referentie embedding van een schoon zonnepaneel.

Deze afstand is een maat voor hoe sterk het huidige beeld afwijkt van een schoon paneel. De gemeten afstand wordt vervolgens genormaliseerd op basis van vooraf bepaalde minimum en maximumwaarden. Op deze manier ontstaat een score tussen 0 en 100 die aangeeft hoe vuil het paneel is.

Op basis van deze score wordt bepaald of het zonnepaneel als schoon of vervuild wordt beschouwd. Daarnaast wordt het niveau van vervuiling ingedeeld in categorieën zoals licht, matig of zwaar vervuild. Deze informatie wordt als uitvoer gebruikt door het systeem om te bepalen of extra schoonmaak nodig is.

Gebruikte materialen:

1. Laptop
2. Raspberry pi
3. Camera
4. Python en OpenCV

Resultaten:

Het systeem werkt op de Raspberry Pi. Camerabeelden werden succesvol opgenomen, verwerkt en geanalyseerd. Bij duidelijke vervuiling werd een hoge afwijkingsscore gemeten, terwijl schone panelen een lage score gaven.

Conclusie:

De hypothese wordt bevestigd. Door het model offline te trainen en daarna toe te passen op een Raspberry Pi kan machine learning effectief worden gebruikt in een praktisch schoonmaaksysteem.

Takenlijst

Kevin Kang	*Jivraj Singh*
OpenCV programma schrijven	ML model training schrijven
Robot map maker programma schrijven	ML encoder schrijven
Robot navigatie maker schrijven	ML model trainen
Robot schoonmaak process schrijven	ML model converter schrijven

Verklaring eigen werk

Kevin was voornamelijk verantwoordelijk voor het ontwerp en de implementatie van de robot en de bijbehorende software. Dit omvatte het ontwikkelen van de navigatiestrategie en het schrijven van de code die de bewegingen en logica van de robot aanstuurt. Daarnaast heeft Kevin gewerkt aan het mappen van het zonnepaneel, het bepalen van schoonmaakpaden en het integreren van de detectieresultaten in het gedrag van de robot.

Jivraj was voornamelijk verantwoordelijk voor het onderzoek naar en de ontwikkeling van de kunstmatige intelligentiecomponenten van het systeem. Hij heeft onderzoek gedaan naar verschillende machine learning-methoden en bepaald welke technieken geschikt waren voor deze toepassing. Daarnaast heeft hij gewerkt aan het verzamelen en voorbereiden van de dataset, het trainen van het machine learning-model en het toepassen van Siamese Neural Networks voor het detecteren van vervuiling. Ook het uitwerken van de theoretische achtergrond viel onder zijn taken.

Conclusie

In dit profielwerkstuk is onderzocht hoe machine learning kan worden ingezet om zonnepanelen efficienter schoon te maken. De centrale hypothese was dat machine learning kan worden ingezet om vervuiling visueel te herkennen en op basis daarvan te bepalen waar extra reiniging nodig is.

Uit het onderzoek blijkt dat deze hypothese grotendeels wordt bevestigd. Door het toepassen van machine learning is de robot in staat om vervuiling op zonnepanelen te detecteren en onderscheid te maken tussen schone en vervuilde delen van het oppervlak. Daarnaast kan het systeem de coordinaat noteren, waardoor de robot later terug kan komen voor herreiniging.

Bijlage

All code staan online op Github.com

https://github.com/KamuiSW/MP-SolarBot

Literatuurlijst

about. (n.d.). Retrieved from raspberrypi.org: https://www.raspberrypi.org/about/

Abtin Mahyar, H. M. (2025). *Feature extraction and deep learning*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/chapter/edited-volume/abs/pii/B9780443328183000198

Allen Newell, H. A. (1976). Computer Science as Empirical Inquiry: Symbols and Search. In H. A. Allen Newell, *Computer Science as Empirical Inquiry: Symbols and Search* (pp. 113-126). 1975 ACM Turing Award Lecture.

Alvi, F. (2023, December 13). *Computer Vision and Image Processing: Understanding the Distinction and Interconnection* . Retrieved from OpenCV: https://opencv.org/blog/computer-vision-and-image-processing/

Artificial intelligence and machine learning in finance: A bibliometric review. (2022, October). Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/abs/pii/S0275531922000344

Cass, S. (2024, August 22). *The Top Programming Languages*. Retrieved from IEEE Spectrum: https://spectrum.ieee.org/top-programming-languages-2024

Davis, E. (2025, June 26). *AI in Robotics: How ROS and Python Are Powering Autonomous Machines*. Retrieved from Codingcops: https://codingcops.com/ai-in-robotics/

Deremuk, I. (2024, September 4). *Python vs. Java: Which One to Choose for Your AI Project?* Retrieved from Litslink: https://litslink.com/blog/python-vs-java-which-one-to-choose-for-ai

Dogra, M. (2020, December 13). *Weakly Supervised Learning for Object Localization*. Retrieved from Medium: https://medium.com/analytics-vidhya/weakly-supervised-learning-for-object-localization-4b73d4f4f4a6

Embeddings: Embedding space and static embeddings. (n.d.). Retrieved from developers.google.com: https://developers.google.com/machine-learning/crash-course/embeddings/embedding-space

Frank Dellaert, S. H. (2023). *5.5 Path Planning*. Retrieved from roboticsbook: https://www.roboticsbook.org/S55_diffdrive_planning.html

How to use classification threshold to balance precision and recall. (2025, January 9). Retrieved from EvidentlyAI: https://www.evidentlyai.com/classification-metrics/classification-threshold

Jaiswal, S. (2024, January 4). *What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling*. Retrieved from Datacamp: https://www.datacamp.com/tutorial/normalization-in-machine-learning

Luca, G. D. (2024, August 30). *Euclidean Distance vs Cosine Similarity* . Retrieved from Baeldung: https://www.baeldung.com/cs/euclidean-distance-vs-cosine-similarity

M. Menagadevi, S. D. (2024). *Machine and deep learning approaches for alzheimer disease detection using magnetic resonance images: An updated review*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/abs/pii/S0263224123016640

Maria Tzelepi, P. N. (2022). *Representation learning and retrieval*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/chapter/edited-volume/abs/pii/B9780323857871000154

Mhd Furqan, R. M. (2021). *Determination of The Closest Path Using The Greedy Algorithm* . Departement of Computer Science, Faculty of Science and Technology, Universitas Islam Negeri Sumatera Utara, Indonesia.

Murphy, I. E. (n.d.). *What Is Machine Learning?* Retrieved from Springer Link: https://link.springer.com/chapter/10.1007/978-3-319-18305-3_1

Nwoke, T. (2023, May 30). *Image Embeddings explained*. Retrieved from Picsellia: https://www.picsellia.com/post/image-embeddings-explained

Physics. (n.d.). Retrieved from unity3d.com: https://docs.unity3d.com/Manual/PhysicsSection.html

Rachid, N. N. (2023, December 8). *A Review on Solar Panel Cleaning Systems and Techniques*. Retrieved from MDPI: https://www.mdpi.com/1996-1073/16/24/7960#B32-energies-16-07960

Rajshekhar. (2022, September 8). *Object Detection Series 3.0 : Sliding Window Approach* . Retrieved from Medium: https://medium.com/@rajshekhar_k/object-detection-series-3-0-basic-of-object-detection-196d91550671

Robinson, J. (1949). *On the Hamiltonian Game (A Traveling Salesman Problem).* The RAND Corporation.

Siamese Neural Network . (2021). Retrieved from ScienceDirect: https://www.sciencedirect.com/topics/computer-science/siamese-neural-network#6.%20Conclusion

Simulated Annealing . (n.d.). Retrieved from cs.cmu.edu: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html

Spasojevic, A. (2024, November 12). *Wat is IT-efficiëntie?* Retrieved from phoenixnap: https://phoenixnap.nl/woordenlijst/het-effici%C3%ABntie

Team, L. (2021, June 10). *Edge Detection Using OpenCV*. Retrieved from LearnOpenCV: https://learnopencv.com/edge-detection-using-opencv/

Tiwari. (2022). *Supervised learning: From theory to applications*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/chapter/edited-volume/abs/pii/B9780128240540000265

Trigolos, A. (n.d.). *C++ VS Python For Machine Learning*. Retrieved from elinext: https://www.elinext.com/solutions/ai/machine-learning/trends/cpp-vs-python-for-machine-learning

Understanding Ultrasonic Sensor. (2023, November 20). Retrieved from Medium: https://medium.com/@robotamateur123/understanding-ultrasonic-sensor-e3791f883061

Unity. (n.d.). *Unity Robotics Hub*. Retrieved from Github: https://github.com/Unity-Technologies/Unity-Robotics-Hub

What are convolutional neural networks? . (2025, November 17). Retrieved from IBM: https://www.ibm.com/think/topics/convolutional-neural-networks

What is Heatmap Data Visualization and How to Use It? (2024, June 12). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/data-visualization/what-is-heatmap-data-visualization-and-how-to-use-it/

Yi, M. (2025). *Heatmap: A complete Guide*. Retrieved from Atlassian: https://www.atlassian.com/data/charts/heatmap-complete-guide