Objektgjenkjenning med ABB

Kenneth R. Eikrehagen
Oslo Metropolitan University
Electrical Engineering
Norway, 3048 Drammen
Email: s331475@oslomet.no

Lars-Erik Ulvund
Oslo Metropolitan University
Electrical Engineering
Norway, 0160 Drammen
Email: s@oslomet.no

Alexsander Kristensen
Oslo Metropolitan University
Electrical Engineering
Norway, 0160 oslo
Email: s@oslomet.no

Erik Johansson
Oslo Metropolitan University
Electrical Engineering
Norway, 0160 Oslo
Email: s@oslomet.no

***Sammendrag*—Sammendraget av hele rapporten skal stå her. Skal inneholde litt av introduksjon, metode og konklusjon.**

I. Introduksjon

Dette prosjektet har blitt inspirert av hvordan kunstig intelligens (KI), tingenes internett (IoT) og skytjenester har påvirket robotteknikk, samt hvordan dette kan fortsette å påvirke denne industrien. På leting etter et prosjekt som kunne passe til dette stakk NAO roboten som hører til i laboratoriet for Automatisering, Robotikk og Intelligente Systemer (ARIS-lab) seg frem som et godt utgangspunkt. NAO er konstruert for å skape forståelse om hvordan KI kan brukes i sammenheng med robotteknikk der internett blir brukt som kommunikasjonskanal. To uker inn i prosjektet ble dessverre NAO-roboten diagnostisert til å være defekt, og prosjektet måtte omstilles for å løse en annen oppgave.

Ny prosjektoppgave beholdt omtrent samme fokus. Kombinasjonen KI og robotteknikk med kommunikasjon gjennom et IP-nettverk. En god måte å implementere dette var ved å kombinere et kamera, socket programmering over Transmission Control Protocol (TCP) og en industriell robotarm fra ABB.

Desember 07, 2020

A. NAO prosjekt

I dette prosjektet skulle NAO roboten undersøke synsområdet sitt etter en ball, gå bort til denne ballen, plukke opp ballen og kaste den. Hvis tiden hadde tillatt det skulle den også respondere på stemmekommando.

Prosjektets tre utfordringer:

1) NAO skulle kjenne igjen ballen og gå bort til den
2) Plukke opp ballen og kaste den
3) Reagere på stemmekommando

Dessverre ble roboten diagnostisert til å være defekt to uker inn i prosjektperioden. På dette tidspunktet var første utfordring unnagjort, og en løsning til neste steg var under utarbeidelse. Mye tid gikk med for å finne kilder til hvordan å kunne programmere roboten med Python 2.7 og brukergrensesnittet til Python-NAO-biblioteket (naoqi). Biblioteket naoqi hadde mange gode funksjoner som kunne brukes i prosjektet, med større fokus på robotteknikk enn KI algoritmer. Koden som ble produsert under dette prosjektet finner man i (referer til appendiks)

B. ABB prosjekt

ABB-robotarmen på ARIS-lab er utrustet med et sugekoppverktøy, og det er montert et kamera rett over robotarmen. Problemstillingen å bruke objektgjenkjenning sammen med robotarmen til å sortere forskjellige geometriske figurer ble raskt unnfanget. De viktigste utfordringene identifisert med dette prosjektet var:

1) Finne en måte å bruke dataen fra kameraet i taket
2) Detektere og finne forskjellige geometriske figurer
3) Få robotarmen til å bevege seg til riktig figur

Videre undersøkelser viste at den beste løsningen var å bruke Python 3.8 (heretter referert til som Python) sammen med OpenCV-biblioteket til å anvende kameraet i taket for objektgjenkjenning. Programvaremessig er Python og RobotStudio brukt for å bevege robotarm; med socket programmering som kommunikasjonsledd. De geometriske figurene som ble brukt for å teste algoritmen var trekant, firkant, sekskant og sirkel.

II. Teori

A. NAO robot

Aldebaran Robotics har laget en liten "humanoid robot" kalt NAO som er designet for å samhandle med mennesker. Den kan gå, danse, snakke og kjenne igjen fjes, stemme og objekt. Denne roboten har nå kommet til sjette generasjon og blir brukt i undervisning, forskning og helsevesen [1]. Den har 25 grader av frihet, 7 berøringssensorer, 4 retningsbestemte mikrofoner og høyttalere, stemmegjenkjenning, 2 2D kameraer og en åpen og fullt programmerbar plattform [2].

B. Objektgjenkjenning

Objektgjenkjenning er en datateknologi som benytter datasyn og bildebehandling for å oppdage objekter av en bestemt klasse, f.eks. mennesker, biler eller sykler i digitale bilder og videoer. For å gjenkjenne objekter benyttes det maskinlæring.

a) Maskin læring: Prosessen med å bruke matematiske datamodeller til å hjelpe en datamaskin med å lære uten direkte instruksjon kalles maskinlæring (ML) [**?**]. Dette kan være "enklere" tilnærming som baserer seg på forhåndsdefinerte egenskaper eller "kjennetegn" som sier noe om hva det letes etter og hvordan det skal finnes, hvor dataen etterpå blir klassifisert ved å bruke f.eks. SVM (Support Vector Machine).

C. OpenCV

Et programvarebibliotek som ofte blir brukt for konstruksjon av kildekode for maskinlæring og datamaskinsyn er OpenCV. Biblioteket er utviklet for å tilby en felles infrastruktur for datasynapplikasjoner og for å akselerere bruken av maskinoppfattelse i kommersielle produkter. OpenCV sitt grensesnitt inkluderer C++, Python, Java og MATLAB og inneholder over 2500 algoritmer innenfor maskinlæring og datasyn [**?**].

D. ABB robot arm

ABB robotarm IRB 140 er en relativ liten, men kraftig robotarm som jobber med seks akser. Robotarmen har en rekkevidde på 810mm til akse fem og en løftekapasitet på maks 6 kg. Det finnes nesten ingen begrensninger for hvor mange forskjellige verktøy som kan monteres på enden av robotarmen, deriblant en sugekopp. Basen til robotarmen kan monteres enten til bakken/underlaget, i taket eller til veggen, noe som gjør robotarmen veldig allsidig [**?**].

a) Robotstudio: Robotarmen styres av programmer som er laget i ABB sitt program RobotStudio. Programmet overføres til en såkalt FlexPendant, som er koblet til og styrer robotarmen. RobotStudio har en innebygget funksjons som gjør det mulig å simulere ønsket program før det testes på den fysiske roboten.

I RobotStudio er det mulig å modellere gjenstander, figurer og workobjects. En løsning som gjør at simuleringer i programmet enkelt kan gjenspeile virkeligheten. Selve programmet skrives i RAPID, et høynivå programmeringsspråk utviklet for ABB sine industrielle roboter. Programmet bygges opp av flere moduler for å oppnå god struktur. RAPID har en rekke interne biblioteker og funksjoner som gjør prosessen med å konstruere forskjellige programmer relativ enkel [**?**] [**?**].

E. Socket programmering

Socket kommunikasjon brukes for å få to noder i et nettverk til å kommunisere med hverandre ved hjelp av meldinger eller pakker. Det kan være et logisk, internt nettverk i en datamaskin, eller et som er fysisk tilkoblet et åpent, ytre nettverk. Det er mulig å sende og motta slike meldinger nesten hvor som helst så lenge man har en internettforbindelse.

Nodene defineres som tjener og klient hvor den ene noden (tjeneren) lytter på en bestemt port – den andre (klienten) når ut for å etablere en tilkobling.

Kommunikasjonsprotokollen som normalt brukes er TCP, hvor man har god kontroll på at meldingene mottas i den formen de ble sendt. Mottakeren kan bekrefte meldingen – og pakker som ikke når frem overføres på nytt av avsender.

a) Transport Control Protocol (TCP): Både tjener og klient etablerer socket. Tjeneren binder socket-en til spesifisert adresse og port, og lytter passivt etter klientens forsøk på tilkobling. Når tjeneren aksepterer en innkommende forespørsel om tilkobling gjøres et treveis håndtrykk for å bekrefte at kommunikasjonen er etablert. Meldinger kan så sendes fritt mellom nodene. Klientens socket lukkes ofte etter utført operasjon, slik at tjeneren kan gå tilbake til passiv lyttemodus, men dette avhenger litt av hensikt og applikasjon.

III. Metode

I denne prosjektoppgaven er objektgjenkjenning sammen med ABB-robotarmen brukt til å sortere forskjellige geometriske figurer. OpenCV og Python-programmering er benyttet for gjennomføring av objektgjenkjenning. ABB-robotarm med sugekopp-verktøy er anvendt for den fysiske sorteringen av figurene. Hvordan robotarmen skulle oppføre seg ble programmert gjennom RAPID i RobotStudio. For å etablere kommunikasjon mellom Python og RobotStudio over internett måtte det benyttes socket programmering.

Algoritmen som ble konstruert ble testet med at robotarmen skulle sortere 4 forskjellige geometriske figurer. Figurene bestod av trekant, firkant, sirkel og sekskant. For enkelheten sin skyld ble benevnelsen forkortet til respektive TRI, SQR, CRC og HEX for videre kommunikasjon med robotarmen. For utstyrsliste se appendiks.

A. ABB-robotarm

Til å sortere de ulike geometriske figurene er det benyttet en ABB-robotarm, og programmet som kjøres på FlexPendant er skrevet i RAPID. Robotarmen ble utstyrt med et sugekopp-verktøy som flyttet de forskjellige figurene fra synsfeltet til kameraet og over til en predefinert "endestasjon".

For å verifisere arbeidsområdet til robotarmen ble det plasserte et hvit ark på bordet foran robotarmen. Kameraet i taket ble skrudd på og prosjektert på en pc-skjerm. En firkant figur ble plassert i hvert hjørnene i henhold til hva kameraet kunne detektere og hjørnene ble markert på det hvite arket.

B. Objektgjenkjenning

OpenCV ble brukt sammen med Python for å kunne fange og prosessere video av ønsket arbeidsområde. For å kommunisere med kamera i taket var det også nødvendig å installere en ekstern driver [**?**], da dette kameraet ikke var "plug and play".

Det som kjennetegner en geometrisk figur er hvor mange kanter den har. OpenCV har en funksjon som kan detektere antall kanter til objekter i bildet. Dette kalles "edge detection" og er en gren som tilhører objektgjenkjenning. Denne funksjonen utførte først objektlokalisering for å finne figurene, deretter ble det telt opp hvor mange kanter figurene hadde. Antall kanter ble deretter analysert for å kjenne igjen hvilke typer geometriske figurer den så på bildet. Ut ifra dette kunne man også kalkulere senterpunktet til hver figur. De foreløpige verdiene måtte deretter skaleres fra pikseldimensjon til metriske verdier. Dette ble gjort ved å beregne forholdet mellom høyden i bildet med tilsvarende høyde på arbeidsområdet. I forbindelse med videre kommunikasjon til robotarmen var det nødvendig å etablere et referansepunkt. Det viste seg at OpenCV analyserte bildet som en matrise ved at den startet i øverste venstre hjørnet, og jobbet seg bortover rad for rad. Referansen for videre kommunikasjon måtte derfor være øverst til venstre i bildet.

C. Socket programmering

For å oppnå kommunikasjon mellom Python koden som tok seg av objektgjenkjenningen og RobotStudio som beveger robotarmen ble det benyttet socket programmering. Python programmet ble satt opp til å være en klient, og robotarmen ble konfigurert til å være server. Før dataen kunne sendes videre til robotarmen måtte alle numeriske verdier konverteres til datatypen "string". Dette ble gjort i en egendefinert funksjon som tok i mot antall identifiserte kanter, samt piksel koordinater til senter i figurene og returnerte dette som en string. Beskjeden som ble sendt inneholdt benevnelsen av hvilken type figur samt senter koordinatene til figuren.

D. RobotStudio

For at ABB robotarmen skulle plukke opp figurene var det viktig å synkronisere arbeidsområdet til kameraet med arbeidsområdet til robotarmen. Kameraet var montert slik at på videoen som ble vist var hjørnet oppe til venstre for kameraet tilsvarende hjørnet nede til høyre for basen til robotarmen. Programmet som ble laget i Python sendte koordinater med utgangspunkt fra hjørnet oppe til venstre. Der X-aksen strakk seg langs øvre kant og Y-aksen nedover langs bildets venstre kant.

Løsningen som ble valgt var å lage et workobject i RobotStudio som tilsvarer arbeidsområdet til kameraet. Arbeidsområdets vinkel i forhold til arbeidsbordet ble justert ved hjelp av en trigonometrisk funksjon, vinkel $A = tan^{-1}\frac{b}{a}$.

Siden kameraet er montert i taket oppfatter den figurene kun i 2D. Det vil si at kameraet ikke kan detektere høydene på figurene som vises i arbeidsområdet. Høyden på de forskjellige figurene må derfor pre defineres for hver figur.

For å unngå at de sorterte geometriske figurene ikke skulle forveksles som figurer som skulle sorteres. Ble "endestasjonene" satt til å være utenfor synsfeltet til kameraet.

E. Figurene

For å vise at prosjektet med objektgjenkjenning kunne kjenne igjen forskjellige geometriske figurer falt valget ned på 4 forskjellig figurer. Hver med forskjellig høyde. En trekant med en høyde på 20 mm, firkant med høyde 15 mm, sekskant med høyde 10 mm og sirkel med høyde 24 mm. De geometriske figurene firkant, trekant og sekskant ble tegnet i Tinkercad [**?**] og 3D-printet med bruk av PLA. PLA er et termoplastisk materiale som er relativt lufttett. Dette gjør det lettere for sugekopp-verktøyet til robotarmen å plukke opp de geometriske figurene. Sirkelen som er benyttet er en gammel snusboks som er laget av plast. PLA ble valgt fordi det var gratis og tilgjengelig på MakerSpace.

IV. Resultater

Resultatene vi oppnådde under dette prosjektet

V. Diskusjon

Hva kunne vi gjort annerledes eller bedre?

VI. Konklusjon

Hva lærte vi av dette prosjektet.

Referanser

[1] IEEE, "Nao." https://robots.ieee.org/robots/nao/. Accessed 02-12-2020.

[2] SoftBanks, "Nao^6." https://www.softbankrobotics.com/emea/en/nao. Accessed 02-12-2020.