Not-Colorblind Nao Robot

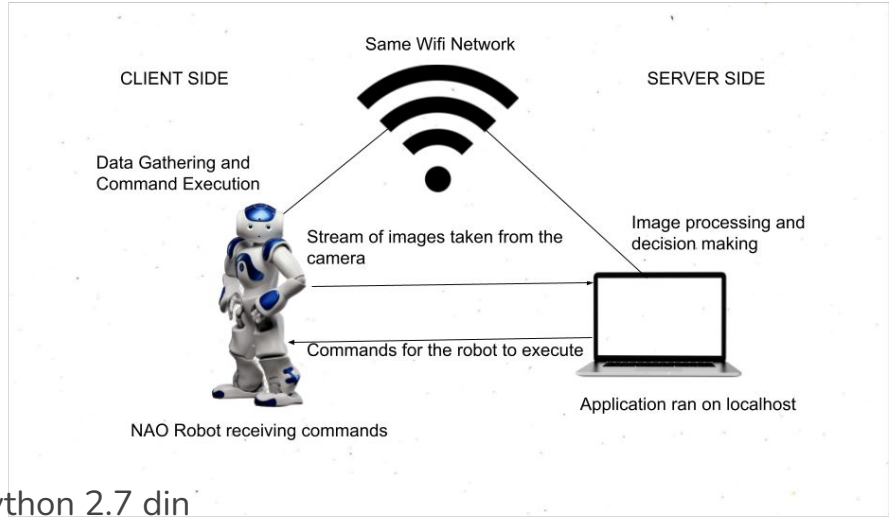
Luka Popovici
Ionuț Ionescu

Context si Motivație

- **Context:** Detectarea obiectului de interes dintr-o imagine și identificarea culorii sale de către robotul NAOO
- **Motivație:** Lucrul cu roboți/Domeniul roboticii
- **Obiectivul proiectului:** Reușirea sortării obiectelor de către robot

Arhitectura preliminara a solutiei

Arhitectura server-client



Mențiune: "Clientul" e o altă aplicație care rulează în Python 2.7 din cauza faptului că nu se poate rula cod scris de direct pe robot. Dar tot vor fi două aplicații care vor comunica prin sockets: Server-side care face interpretarea și va rula pe ultima versiune Python și Client-side care va fi rulat pe Python 2.7 pentru a apela funcții NAOqi și va face comunicarea cu robotul. În scope-ul diagramei, clientul este robotul, deși aplicația Client funcțional va avea rol de interfață laptop - robot.

Componente

- Robotul NAO (titular) - device-ul care produce imaginile ce vor fi procesate. Comunica cu aplicația client prin internet cu ajutorul API-ului NAOqi.
- Aplicația client care preia imaginea de la robot si o va da serverului sa fie procesata. Odata ce culoarea obiectului de interes din imagine e determinata, serverul transmite clientului rezultatul, iar acesta la rândul lui va transmite instrucțiuni robotului pentru a identifica prin Text-To-Speech culoarea.
- Aplicația server e cea responsabila de interpretarea imaginii.

FLUXUL DE DATE

Robot Nao -(internet/ prin intermediul NaoQI)-> Client [Imagine]

Client -(localhost/sockets)-> Server [Imagine]

Server -(localhost/sockets)-> Client [Response]

Client -(internet/ prin intermediul NaoQI)-> Robot Nao [Response]

In ultima etapa se verbalizeaza response-ul.

Evaluarea solutiei si setul de date folosit

Ca set de date am folosit mai multe obiecte colorate cu contrast mare între ele si fundal. Pozele pe care le-am folosit au fost selectate în acest fel pentru a ajuta algoritmul de edge detection.

Metrica de evaluare constă în capacitatea alogoritmului de a selecta o porțiune din obiectul de interes din imagine și dacă estimează în mod corect culoarea acestuia.

ALGORITMUL FOLOSIT

Algoritmul poate fi separat in 2 etape: etapa de Preprocesare si cea de Procesare.

Etapa de Preprocesare

1. Presegmentare:

Aceasta implică divizarea imaginii în regiuni semnificative, in acest caz o folosim pentru a ne concentra pe centru.

2. Resize:

Redimensionarea imaginii pentru a se pentru garanta un timp mai constant in aplicarea agoritmilor.

3. Conversie Grayscale:

Transformarea imaginii color în tonuri de gri, ceea ce simplifică procesarea și reduce complexitatea datelor.

Etapa de Preprocesare

4. Egalizare de Histogramă:

Această tehnică îmbunătățește contrastul imaginii prin redistribuirea valorilor pixelilor, făcând detaliile mai vizibile.

5. Gaussian Blur:

Aplicarea unui filtru gaussian pentru a reduce zgomotul și a netezi imaginea, ajutând la o detecție mai precisă a contururilor.

6. Adaptive Threshold Gaussian:

Această metodă determină pragurile localizate pentru binarizarea imaginii, adaptându-se la variațiile de iluminare din imagine.

Etapa de Preprocesare

7.Morphological Closing:

Utilizarea operațiunilor morfologice pentru a umple golurile din contururi și a conecta regiuni apropiate, îmbunătățind integritatea formelor detectate.

8. Aplicare algoritm Canny pentru detecția muchiilor:

Algoritmul Canny este utilizat pentru a identifica marginile obiectelor din imagine.

Etapa de Procesare

1. Selectare contur lungime maximă:

Identificarea și selectarea contururilor cu lungimea maximă din imagine, care sunt relevante pentru analiza ulterioară.

2. Generarea mască și decupare obiect:

Crearea unei măști bazate pe contururile selectate și decuparea obiectului de interes din imagine.

Etapa de Procesare

3. Calcularea mediei aritmetice a pixelilor conturului de interes:

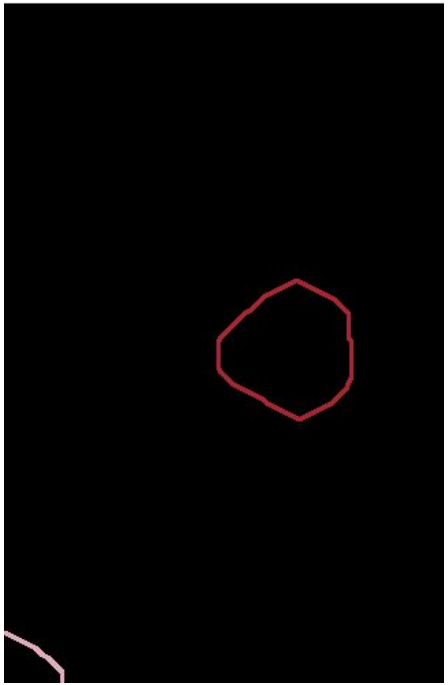
Determinarea valorii medii a pixelilor din interiorul conturului selectat.

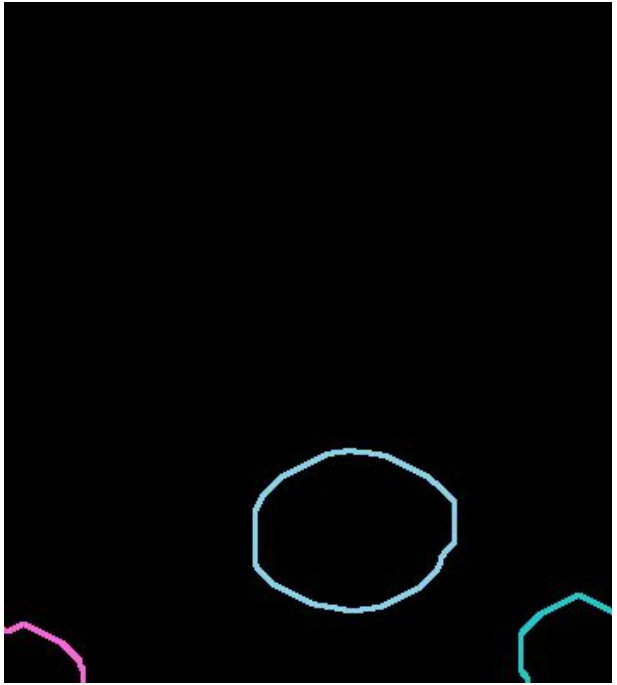
4. Încadrarea valorilor pentru fiecare canal în niște praguri și afișarea rezultatelor:

Stabilirea unor limite pentru valorile pixelilor în fiecare canal de culoare (de exemplu, R, G, B) și prezentarea rezultatelor într-un format clar și concis.

Imaginile Originale

Preprocesare - Vizualizarea contururilor



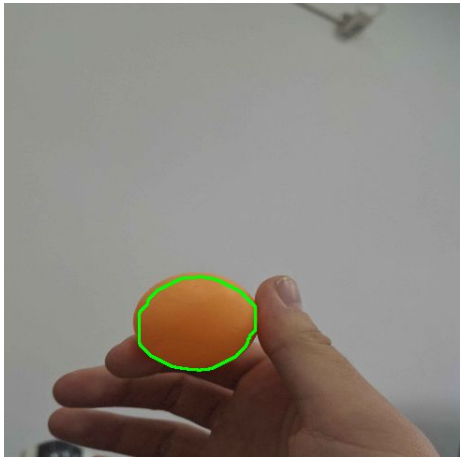


Rezultate

Pixeli selectati conform mastii obtinute in urma preprocesarii

Color: Red

Color: Light Grey



Color: Orange

Constrângeri

Limitarile solutiei actuale:

Nu pot sa fie contururi foarte proeminente in fundal.

Contrastul dintre obiectul de interes și fundal trebuie sa fie proeminent

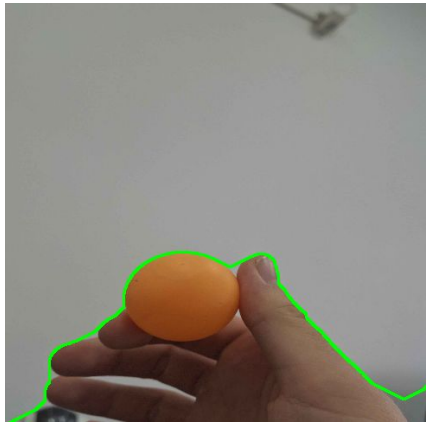
Incadrarea in praguri exacte ale culorilor

REZUMAT

Pe imaginile care îndeplinesc criteriile menționate în slide-ul anterior s-a detectat in mod consistent culoarea corecta.

PROBLMA PRINCIPALA: PREA MULTE EDGE-URI!

Dilema: Exista șanse ca muchiile detectate să fie luate în mode separat, iar unirea lor agresivă cu algoritmul Canny duce la un blend-in cu alte forme, care vor da un rezultat eronat.



Ajustări

Potentiale imbunatatiri:

Folosirea unor modele de machine learning pentru detectarea contururilor/ pentru acuratete mai mare a incadrarii culorilor.

Direcții viitoare

Pasii urmatori: Vom realiza comunicarea cu robotul NAO / posibil implementarea unui Model de ML cum a fost mentionat mai sus.

Mod de implementare: Algoritmul de prelucrare este modular si pot fi create usor noi funcții care să comunice cu componentele deja existente.

Obiectivele finale: Să avem un robot NAO funcțional care să indentifice corect obiectele care îi sunt arătate.