Scanarea și rezolvarea unui cub Rubik utilizând tehnici de Machine Vision
Procesarea Imaginilor

Vlad Ursache

1 Introducere

Proiectul prezentat în această documentație se referă la dezvoltarea unui sistem de scanare a celor 6 fețe ale unui cub Rubik folosind tehnici de Machine Vision. Scopul principal al proiectului este detectarea culorilor de pe fiecare față a cubului și encodarea stării acestuia în vederea rezolvării automate.

2 Implementare

Pentru implementarea proiectului s-a folosit limbajul de programare Python și biblioteca OpenCV pentru prelucrarea imaginilor. Algoritmul Kociemba este utilizat pentru rezolvarea cubului Rubik. Codul sursă a fost organizat în funcții și structuri de date care sunt detaliate mai jos:

2.1 Variabile și structuri de date

- `stickers_position`: Dicționar care conține coordonatele pixelilor pentru stickerele plasate strategic pe ecran.
- `color`: Dicționar care mapează denumiri de culori la valori RGB.
- `check_state`: Listă care stochează stările verificate ale cubului.
- `solution`: Listă care va conține soluția pentru rezolvarea cubului.
- `solved`: Variabilă booleană care indică dacă cubul a fost rezolvat sau nu.
- `state`: Dicționar care stochează starea curentă a fiecărei fețe a cubului.
- `sign_conv`: Dicționar care mapează denumiri de culori la semne corespunzătoare pentru algoritmul Kociemba.

2.2 Funcții principale

- `draw_stickers`: Funcție care desenează stickerele pe imaginea capturată.
- `color_detect`: Funcție care detectează culoarea în funcție de valorile H, S și V din modelul de culoare HSV.
- `solve`: Funcție care rezolvă cubul Rubik folosind algoritmul Kociemba și returnează soluția.
- `main`: Funcția principală care inițializează camera, capturează și prelucrează imaginile și gestionează interacțiunea cu utilizatorul.

3 Fluxul de lucru

1. Inițializarea camerelor și a parametrilor.
2. Capturarea imaginii.
3. Convertirea imaginii din spațiul de culoare BGR în HSV.
4. Detectarea stickerelelor și extragerea valorilor de culoare corespunzătoare.
5. Actualizarea stării cubului pe baza valorilor de culoare detectate.
6. Verificarea stării cubului și, în cazul completării, rezolvarea cubului.
7. Afișarea imaginii capturate cu stickerele colorate și soluția cubului (dacă este disponibilă).
8. Repetarea pașilor 2-7 până când utilizatorul întrerupe programul.

4 Utilizare

Pentru a utiliza acest program, urmați pașii de mai jos:

1. Asigurați-vă că aveți Python și biblioteca OpenCV instalate pe sistemul dumneavoastră.
2. Asigurați-vă că aveți o cameră video conectată la calculator.
3. Executați scriptul Python care conține codul sursă.
4. Așteptați ca programul să deschidă interfața camerei video.
5. Plasați cubul Rubik în fața camerei, astfel încât stickerele să fie vizibile.
6. Apăsați tastele corespunzătoare pentru a înregistra stările cubului.
7. După ce ați înregistrat stările pentru toate cele 6 fețe, apăsați tasta Enter pentru a obține soluția.
8. Soluția cubului va fi afișată în consolă.

5 Algoritmul în două faze al lui Kociemba

Algoritmul este eficient pentru rezolvarea cubului Rubik și este utilizat în proiectul nostru pentru a găsi soluția cubului pe baza stării detectate. Algoritmul este dezvoltat de Herbert Kociemba și utilizează o abordare în două etape pentru a rezolva cubul Rubik.

5.1 Etapa 1: Construirea arborelui de căutare

În prima etapă a algoritmului, se construiește un arbore de căutare pentru a explora toate posibilitățile de mutări ale cubului și a găsi cea mai scurtă soluție. Algoritmul folosește o strategie de căutare euristică bazată pe distanța Manhattan pentru a evalua starea cubului în funcție de diferența față de configurația scop.
Pentru a construi arborele de căutare, algoritmul generează și explorează recursiv toate configurațiile posibile ale cubului prin aplicarea diferitelor secvențe de mișcări (rotații) pe cub. Fiecare nod al arborelui reprezintă o configurație a cubului, iar muchiile reprezintă mișcările care transformă o configurație în alta. Se utilizează o strategie de tăiere a ramurilor care nu conduc la o soluție mai scurtă pentru a reduce timpul de căutare.

5.2 Etapa 2: Rezolvarea efectivă a cubului

În etapa a doua a algoritmului, se utilizează rezultatele primei etape pentru a găsi soluția efectivă a cubului. Algoritmul aplică o căutare inversă pornind de la configurația scop și utilizând mișcările inverse pentru a ajunge la starea inițială a cubului. Acest lucru se realizează prin urmărirea drumului de la configurația scop către configurația inițială în arborele de căutare generat în etapa anterioară. Pentru a reduce timpul de căutare, se utilizează o tehnică de tăiere a ramurilor care nu conduc la configurația inițială. Astfel, algoritmul găsește soluția cubului într-un timp rezonabil, evitând explorarea inutilă a unor configurații irelevante.

5.3 Implementarea algoritmului în două faze al lui Kociemba

În proiectul nostru, implementarea algoritmului în două faze al lui Kociemba este realizată prin intermediul bibliotecii Kociemba, care oferă funcționalități pentru rezolvarea cubului Rubik. După ce obținem starea cubului pe baza culorilor detectate în etapa de scanare, utilizăm algoritmul Kociemba pentru a găsi soluția cubului.
Folosind funcția `kociemba.solve()` din biblioteca Kociemba, furnizăm configurația cubului sub forma unui șir de semne corespunzătoare culorilor detectate. Algoritmul returnează apoi soluția cubului, care este afișată în consolă și poate fi utilizată în continuare pentru a rezolva cubul Rubik.

5.4 Avantaje și limitări

Algoritmul în două faze al lui Kociemba are câteva avantaje importante. Este un algoritm eficient și rapid, capabil să găsească soluția cubului Rubik într-un timp rezonabil. De asemenea, abordarea în două etape și utilizarea unei strategii de căutare euristică permit reducerea spațiului de căutare și obținerea unei soluții optime.
Cu toate acestea, algoritmul are și câteva limitări. Una dintre ele este că este optimizat pentru rezolvarea cubului în cel mult 20 de mișcări, ceea ce înseamnă că nu întotdeauna găsește cea mai scurtă soluție posibilă. De asemenea, algoritmul poate fi sensibil la erorile sau incertitudinile în detectarea culorilor cubului, ceea ce poate duce la soluții incorecte sau incomplete.

6 Concluzii

Proiectul de Machine Vision prezentat în această documentație demonstrează aplicarea conceptelor de prelucrare a imaginilor pentru a detecta și a recunoaște culorile de pe fețele unui cub Rubik. Prin utilizarea algoritmului Kociemba, programul poate rezolva cubul Rubik pe baza stării detectate.