1. **Funcția Canny**

Utilizări:

Funcția Canny este utilizată într-o gamă largă de aplicații care necesită detectarea și extragerea marginilor din imagini, cum ar fi:

-Segmentarea imaginilor.

-Recunoașterea obiectelor.

-Detectarea contururilor.

-Procesarea și analiza imaginilor medicale.

-Procesarea imaginilor pentru aplicații de realitate augmentată și computer graphics.

Algoritm:

Algoritmul Canny este un algoritm de detecție a marginilor în două etape, care constă în:

a)Calculul gradientului: Se utilizează operații de convoluție cu kerneluri Sobel pentru a calcula magnitudinea gradientului și direcția acestuia în imaginea sursă.

b)Suprimarea non-maximelor și histeresisul: Se elimină non-maximele (punctele care nu sunt maxime locale pe direcția gradientului) și se aplică un proces de histeresis pentru a conecta și filtra marginile.

Funcția returnează o imagine binară în care pixelii de margini sunt marcați cu valori non-zero (alb), iar restul imaginii este setată la zero (negru).

Poate avea mai mulți parametrii precum:

a)image: Este imaginea sursă în care se vor detecta marginile. Această imagine trebuie să fie o imagine în tonuri de gri (deși, în practică, funcția poate fi aplicată și pe imagini color).

b)threshold1 și threshold2: Acești doi parametri reprezintă pragurile minim și maxim pentru detectarea marginilor. Aceste praguri controlează cât de pronunțate trebuie să fie marginile detectate pentru a fi considerate valide.

c)apertureSize: Este dimensiunea kernelului Sobel utilizată pentru a calcula gradientul. Acesta specifică dimensiunea fereastră utilizată pentru calcularea gradientului, care la rândul său afectează sensibilitatea la zgomot a algoritmului.

-Valoarea implicită este 3, ceea ce indică utilizarea unui kernel de 3x3 pentru calculul gradientului.

-Dimensiunea kernelului trebuie să fie un număr impar.

d)L2gradient: Un parametru boolean care indică dacă să se folosească norma L2 (True) sau norma L1 (False) pentru calculul gradientului. Acest lucru afectează formula de calcul al magnitudinii gradientului. Valorile implicite este False, ceea ce indică utilizarea normei L1.

1. **Funcția VideoCapture**

Prin utilizarea funcției VideoCapture() din biblioteca OpenCV, este foarte ușor să capturăm un flux video în timp real de la o cameră pe fereastra OpenCV.

Această funcție necesită un indice de dispozitiv ca parametru. Calculatorul poate avea mai multe camere atașate. Acestea sunt enumerate printr-un indice care începe de la 0 pentru camera web încorporată. Funcția returnează un obiect VideoCapture. Atunci când apelăm această funcție cu un anumit indice de dispozitiv, vom putea captura fluxul video de la respectiva cameră și să-l procesăm sau să-l afișăm în fereastra OpenCV, oferindu-ne astfel posibilitatea de a lucra cu fluxuri video în timp real în aplicațiile noastre de prelucrare a imaginilor sau de visionare.

1. **Functia CapRead()**

Metoda cap.read() este utilizată pentru a citi un singur cadru dintr-un flux video. Această metodă este aplicabilă obiectului VideoCapture pe care l-am creat anterior folosind funcția cv.VideoCapture(). Când apelăm metoda cap.read(), ea returnează:

-un rezultat boolean (ret): Acest rezultat indică dacă citirea cadrelor a fost realizată cu succes sau nu. Dacă citirea a fost realizată cu succes, valoarea este True; dacă nu, valoarea este False.

**4. Funcția cv.cvtColor**

Această funcție este utilizată într-o varietate de aplicații de prelucrare a imaginilor, inclusiv în segmentarea obiectelor, extragerea caracteristicilor, detectarea marginilor etc. Convertirea într-un spațiu de culori corespunzător poate face mai ușor procesarea ulterioară a imaginilor în funcție de cerințele specifice ale aplicației.

În acest caz, este folosit pentru a converti fiecare cadru la tonuri de gri.