ORB DESKRIPTORLARI YORDAMIDA YUZNI TANIB OLISH DASTURIY TA'MINOTINI ISHLAB CHIQISH

Abdumalikov Abduqodir Abdug'affor o'g'li

Andijon davlat universiteti, Axborot texnologiyalari kafedrasi o'qituvchisi.

Abduqodircoder@gmal.com

Annotatsiya.Bu maqolada ORB deskriptorlari muhimligi. ORB deskriptoridan foydalanib python 3.9.6 opencv-python 3.4.11.39 vositasida dastur tuzish ko'rib o'tilgan.

Kalit so'zlar: BRIEF, ORB, BRISK, deskriptor, mahalliy deskriptorlar,

Abstract:

The importance of ORB descriptors in this article. Python 3.9.6 opencv-python 3.4.11.39 program creation using the ORB descriptor is considered.

Keywords:

BRIEF, ORB, BRISK, descriptor, local descriptors.

Аннотация:

Важность дескрипторов ORB в этой статье. Рассмотрено создание программы на Python 3.9.6 opencv-python 3.4.11.39 с использованием дескриптора ORB.

Ключевые слова:

BRIEF, ORB, BRISK, дескриптор, локальные дескрипторы

Deskriptor - bu xususiyatlar to'plamiga asoslangan holda tasvirning ba'zi bir mintaqasini aniqlaydigan usul. Ko'pincha tasvirlardagi obyektlarni izlash deskriptorlar yordamida solishtirishga asoslanadi. Ikki o'lchovli tasvir deskriptorlarining quyidagi guruhlari ajratiladi gradientga asoslangan deskriptorlar, lokal ikkilik deskriptorlar, spektral tasvirga asoslangan deskriptorlar, bazis funksiyali deskriptorlar, shakl deskriptorlari. Shuni ta'kidlash kerakki, ba'zi usullar, ularning xususiyatlariga ko'ra, bir vaqtning o'zida turli guruhlarga bo'linishi mumkin.

Mahalliy ikkilik identifikatorlar ikkilik vektorlar ko'rinishidagi tasvirning kichik maydonining tavsifidir. Eng mashhur mahalliy deskriptorlar mahalliy binar naqshlar va ularning modifikatsiyalari (Local Binary Patterns - LBP). Bu guruhga shuningdek BRIEF (Binary Robust Independent Elementary Features) , ORB (Oriented BRIEF), BRISK (Binary Robust Invariant Scalable Keypoints) va boshqalar kiradi. ORB BRIEF deskriptorining takomillashtirilgan versiyasi bo‘lib, uning ayrim kamchiliklarini hal qiladi. Maxsus nuqtalarni topish uchun u tasvirdagi piksellarni tasniflash uchun qarorlar daraxtlarini quruvchi FAST detektoridan foydalanadi. ORB ning asosiy xususiyatlari:

• FASTga tez va aniq orientatsiya komponentini qo'shish;

• QISQA yo'naltirilgan xususiyatlarni samarali hisoblash;

• dispersiya va korrelyatsiyani tahlil qilish, QISQA yo'naltirilgan xususiyatlar;

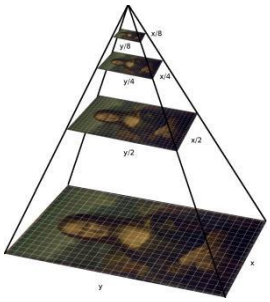
• aylanish o'zgarmasligi ostida QISQA funksiyalarning dekorrelyatsiyasini o'rganish usuli, bu esa samaradorlikni oshirishga olib keladi. Koordinatalari (x, y) bo'lgan n o'lchamdagi ikkilik testlar to'plami uchun 2xn matritsa S quriladi:

$$S = \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix}$$

1-rasm. 2xn S matritsa

FAST funksiyalari orientatsiya komponentiga ega emas va bir nechta masshtab funksiyalariga ega emas. Shunday qilib, ORB algoritmi ko'p masshtabli tasvir piramidasidan foydalanadi. Tasvir piramidasi - bu bitta tasvirning ko'p masshtabli tasviri bo'lib, u tasvirlar ketma-ketligidan iborat bo'lib, ularning har biri asl tasvirning qisqartirilgan aniqlikdagi versiyasidir. Piramidaning har bir darajasi oldingi darajaga qaraganda tasvirning kichikroq versiyasini o'z ichiga oladi. ORB tasvir piramidasini yaratgandan so'ng, tasvirdagi asosiy nuqtalarni aniqlash uchun FAST algoritmidan foydalanadi. Piramidaning har bir darajasidagi asosiy

nuqtalarni aniqlash orqali ORB bitta tasvirning turli masshtablarida asosiy nuqtalarni samarali tarzda topadi. Piramidaga misol quyidagi 2-rasmda keltirilgan.

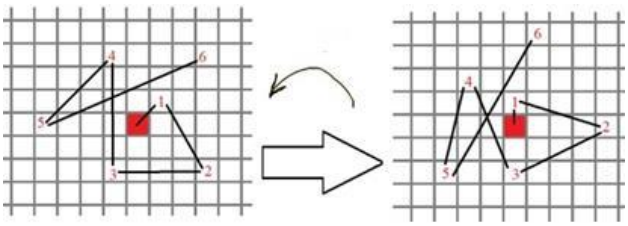


2-rasm – Tasvir piramidasi

Asosiy nuqtalar topilgandan so'ng, ORB har bir asosiy nuqtaga, masalan, chap yoki o'ngga, ushbu asosiy nuqta atrofida intensivlik darajasi qanday o'zgarishiga qarab, yo'nalishni belgilaydi. ORB intensivlik o'zgarishini aniqlash uchun intensivlik markazidan foydalanadi. Intensivlik markazi burchakning intensivligi uning markazidan siljishini taxmin qiladi va bu vektor orientatsiyani aniqlash uchun ishlatilishi mumkin. ORB usuli burchak orientatsiyasini hisoblash uchun tortishish markazining koordinatalaridan foydalanadi, yamoq m momentlari orqali hisoblanadi:

$$m_{pq}=\sum_{x,y} x^p y^q I(x,y)$$

Keyin burchakning yo'nalishi vektor tomonidan belgilanadi, uning boshlanishi markaziy nuqtada va oxiri og'irlik markazida bo'ladi. Yagona nuqtalarning aylanishiga misol quyidagi 3-rasmda keltirilgan.



3-rasm - Nuqtalar ketma-ketligini 30 gradusga aylantirish

Deskriptorlarning sifatini keyingi taqqoslash asosida aniqlash ko'rib chiqiladi. Ba'zi hollarda bitta rangli sohadagi deskriptor o'sha nuqtada boshqa identifikatorlar bilan o'zaro bog'liq bo'lib, farqlash qiyin bo'ladi. Shu sababli, tavsiflovchining samaradorligi o'rtacha arifmetik va dispersiya parametrlari orqali baholanadi. ORB deskriptorida katta tafovut mavjud va o'rtacha qiymat taxminan 0,5 ga teng. Biroq, asosiy nuqtaning yo'nalishini topgandan so'ng, deskriptorning dispersiyasi sezilarli darajada kamayadi. ORB deskriptori bu muammoni barcha mumkin bo'lgan ikkilik testlarni solishtirish orqali hal qiladi. Ushbu taqqoslashlar asosida maksimal va minimal dispersiya qiymatlari, shuningdek, 0,5 ga yaqin qiymatlar topiladi. Ikkilik testlarni solishtirish uchun ko'p o'lchovli ma'lumotlarning o'lchamlarini kamaytirishning LSH (Locality-sensitive hashing) usuli qo'llaniladi. Shunday qilib, ko'rib chiqilayotgan deskriptorlarning samaradorligini oshirish bo'yicha asosiy say-harakatlarni ikkita yo'nalishda jamlash kerak: hisoblash optimallashtirish va SIFTni tezlashtirish va ORB sifatini yaxshilash.aniqlanadi.

ORB dan foydalanishga doir Python dasturlash tilida tuzilgan dastur.

```
import cv2
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
orb = cv2.ORB_create()
saved_images = [inson1.jpg'," inson2"]
saved_descriptors = []
for image_path in saved_images:
    img = cv2.imread(image_path, 0)
    faces = face_cascade.detectMultiScale(img, 1.3, 5)
    for (x, y, w, h) in faces:
        roi_gray = img[y:y + h, x:x + w]
        keypoints, descriptor = orb.detectAndCompute(roi_gray, None)
```

```
        saved_descriptors.append(descriptor)
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
        roi_gray = gray[y:y + h, x:x + w]
        keypoints, descriptor = orb.detectAndCompute(roi_gray, None)
        if descriptor is not None:
            bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            best_match_index = None
            best_match_count = 0
            for i, saved_descriptor in enumerate(saved_descriptors):
                if saved_descriptor is not None:
                    matches = bf.match(descriptor, saved_descriptor)
                    if len(matches) > best_match_count:
                        best_match_count = len(matches)
                        best_match_index = i
            if best_match_index is not None:
                cv2.putText(frame, saved_images[best_match_index].split(".")[0], (x, y
- 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Dasturda python versiyasi 3.9.6 opencv-python 3.4.11.39 dan foydalanilgan. inson1 va inson2 avvaldan mavjud rasmlar yuzni aniqlash uchun. Yuzni tanib olishda ORB deskriptoridan foydalanib ushbu dasturlardan foydalaniladi.

Foydalanilgan adabiyotlar ro'yxati

1. Rublee, E., Rabaud, V., Ponce, J., & L epetit, V. (2011). "ORB: An Efficient Alternative to SIFT and SURF." 2011 International Conference on Computer Vision (ICCV).

2. Lepetit, V., & Fua, P. (2005). "Keypoint Recognition Using Randomized Trees." IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(9), 1465-1479.

3. Mikolajczyk, K., & Schmid, C. (2005). "A Performance Evaluation of Local Descriptors." IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10), 1615-1630.

4. Alahi, A., Ozuysal, M., & Farinella, G. M. (2012). "FREAK: Fast Retina Keypoint." 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 802-809.

5. Sutherland, D., et al. (2015). "Efficient and Accurate Keypoint Matching using 2D Binary Descriptors." Pattern Recognition, 48(3), 830-841.

6. Hernández, C., & Vázquez, A. (2017). "Comparison of Feature Detectors and Descriptors for Image Matching." Journal of Electronic Imaging, 26(4), 043004.

7. Zhang, Z., & Liu, J. (2020). "A Review of Keypoint Detection and Description Algorithms." Computational Intelligence and Neuroscience.