KOKAND UNIVERSITY HERALD SCIENTIFIC E-JOURNAL

QO‘QON UNIVERSITETI XABARNOMASI
KOKAND UNIVERSITY HERALD
ВЕСТНИК КОКАНДСКОГО УНИВЕРСИТЕТА

ISSN: 2181-1695 **OAK:** 01-08/1819/6 www.herald.kokanduni.uz

C++ DASTURLASH TILIDA MATRITSA ORQALI TASVIR RANGINI ANIQLASH

Boritov Muzaffar Mansurovich
Qo‘qon universiteti, Raqamli texnologiyalari va matematika kafedrasi o‘qituvchi
botirovmuzaffarmansurov@gmail.com

MAQOLA HAQIDA

Qabul qilindi: 24-mart 2025-yil
Tasdiqlandi: 26-mart 2025-yil
Jurnal soni: 14
Maqola raqami: 60
DOI: https://doi.org/10.54613/ku.v14i.1173

KALIT SO‘ZLAR/ КЛЮЧЕВЫЕ СЛОВА/ KEYWORDS

RGB, OpenCV, RGB, iostream, namespace, const.

ANNOTATSIYA

Raqamli tasvirlarni qayta ishlash sohasida tasvirning ranglarini aniqlash va tahlil qilish muhim ahamiyat kasb etadi. Ushbu maqolada C++ dasturlash tilidan foydalanib, tasvir piksellarini matritsa shaklida ifodalash va ularning ranglarini aniqlash usullari ko‘rib chiqiladi. OpenCV kutubxonasi yordamida tasvirni yuklash, uning matritsa shaklidagi ko‘rinishini olish va har bir pikselning rangini aniqlash algoritmlari ishlab chiqiladi. Tadqiqot davomida RGB va HSV rang modellaridan foydalanish samaradorligi baholanadi.

Kirish. Bugungi kunda raqamli texnologiyalar hayotimizning deyarli barcha jabhalariga chuqur kirib kelgan. Ular orasida tasvirlarni avtomatik tarzda tahlil qilish texnologiyalari alohida ahamiyatga ega bo‘lib, ko‘plab sohalarda – tibbiyot, sanoat, xavfsizlik, transport, qishloq xo‘jaligi, ekologiya va sun’iy intellektda – keng qo‘llanilmoqda. Ayniqsa, tasvirlar asosida obyektlarni aniqlash, yuzni tanib olish, harakatni kuzatish, tibbiy diagnostika va atrof-muhit monitoringi kabi amaliy sohalarda bu texnologiyalar samaradorlikni sezilarli darajada oshirgan. Tasvirlarni avtomatik tahlil qilish – bu kompyuter yordamida raqamli tasvirlardagi ma’lumotlarni aniqlash, qayta ishlash, tahlil qilish va ularning asosida qaror qabul qilishni ta’minlovchi texnologiyalar to‘plamidir. Bu jarayonlar sun’iy intellekt va kompyuter ko‘rish (computer vision) tizimlari bilan chambarchas bog‘liq. Kompyuter ko‘rish – bu mashina va dasturiy vositalarga inson kabi tasvir va video ma’lumotlarni “ko‘ra olish” va tushunish imkonini beruvchi soha bo‘lib, u aynan tasvirlar bilan ishlashda katta rol o‘ynaydi. Shu nuqtai nazardan, raqamli tasvirlar bilan ishlashda tasvirni segmentatsiya qilish va undagi ranglarni aniqlash juda muhim hisoblanadi. Segmentatsiya – bu tasvirni mantiqiy qismlarga ajratish bo‘lib, bu orqali tasvirdagi har xil obyektlar, fon, chegaralar va tuzilmalarning aniqlanishi ta’minlanadi. Segmentatsiya ko‘plab usullar yordamida bajariladi, ulardan eng samaralilaridan biri – rangga asoslangan segmentatsiyadir. Ranglar tasvirdagi eng asosiy axborot manbalaridan biri hisoblanadi. Har bir tasvir turli rangdagi piksellardan iborat bo‘lib, ularning RGB (qizil, yashil, ko‘k) yoki boshqa rang modellari (HSV, YCbCr, LAB) orqali ifodalanishi mumkin. Ranglarni to‘g‘ri aniqlash va tahlil qilish yordamida tasvirdagi ma’lumotlarni mantiqiy anglash imkoniyati yaratiladi. Masalan, tibbiyotda MRI yoki ultratovush tasvirlarida ranglar o‘zgarishiga qarab kasallik aniqlanishi mumkin. Yoki qishloq xo‘jaligida o‘simlik barglarining rangi orqali ularning sog‘lig‘i baholanadi.

Tasvirlardagi har bir piksel – bu ma’lum bir rangli nuqtadir. Barcha tasvirlar aslida katta o‘lchamdagi matritsalar (ya’ni ikki yoki uch o‘lchamli massivlar) sifatida tasavvur qilinadi. Matritsa usuli orqali har bir pikselning joylashuvi (x, y koordinatalari) va rang qiymati (masalan, R=255, G=100, B=50) aniqlanadi. Ana shu qiymatlar asosida tahlil algoritmlari yaratiladi. Rangga asoslangan tahlil usullari, xususan, thresholding, color filtering, k-means clustering va boshqa mashina o‘rganish algoritmlari bilan birgalikda ishlatiladi. Bu texnologik jarayonlarni dasturlashda esa samarali, tez va funksional vositalar zarur bo‘ladi. Ulardan eng mashhurlaridan biri – bu OpenCV (Open Source Computer Vision Library) kutubxonasidir. OpenCV – bu ochiq manbali, C++ (va boshqa tillar) asosida yaratilgan kutubxona bo‘lib, u tasvir va video ma’lumotlarini tahlil qilish, ularni filtrdan o‘tkazish, obyektlarni aniqlash, segmentatsiya, yuzni tanish, rang bilan ishlash va boshqa ko‘plab funksiyalarni o‘z ichiga oladi. Uning imkoniyatlari nafaqat oddiy dasturlar, balki sanoat darajasidagi ilovalar uchun ham yetarli. C++ dasturlash tili esa ushbu jarayonni yuqori tezlikda, resurslarni nazorat qilib bajarish imkonini beradi. Ayniqsa, real vaqt rejimida ishlovchi ilovalar (masalan: videokuzatuv tizimlari, robototexnika, avtomatik haydovchi tizimlari) uchun C++ va OpenCV tandemida ishlab chiqilgan algoritmlar samaradorlikda katta ustunlikka ega. Shu sababli ushbu maqolada C++ tilida OpenCV kutubxonasi asosida tasvir ranglarini aniqlash jarayoni, uning algoritmlari va real amaliyotda qanday natijalar berishi atroflicha tahlil qilinadi. Xulosa qilib aytganda, tasvirlarni avtomatik tahlil qilish, xususan ranglarni aniqlash va segmentatsiya qilish zamonaviy IT tizimlarida juda *dolzarb* masala bo‘lib, u ko‘plab amaliy sohalarda qo‘llanilmoqda. Bu jarayonlarni samarali bajarish uchun esa C++ tili va OpenCV kabi kuchli vositalardan foydalanish muhim ahamiyatga ega. Shu jihatdan, ushbu mavzu ilmiy-tadqiqot, ta’lim va amaliy dasturlashda katta o‘rinni egallaydi va o‘z dolzarbligini hech qachon yo‘qotmaydi.

Adabiyotlar tahlili. Tasvirlarni qayta ishlash bo‘yicha quyidagi ilmiy manbalardan foydalanildi

1. Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing – tasvirlarni qayta ishlash asoslari yoritilgan.

2. Bradski, G., & Kaehler, A. (2008). Learning OpenCV – OpenCV yordamida tasvirlarni qayta ishlash metodlari bayon qilingan.

3. Solomon, C., & Breckon, T. (2011). Fundamentals of Digital Image Processing – tasvirlarni segmentatsiya qilish va tahlil qilish usullari bayon etilgan.

Raqamli tasvirlarni qayta ishlash sohasida ko‘plab ilmiy manbalar mavjud bo‘lib, ularning ayrimlari ushbu tadqiqotda asosiy metodologik asos sifatida xizmat qildi. Jumladan, Gonzalez va Woods tomonidan yozilgan Digital Image Processing asarida tasvirlarni raqamli qayta ishlashning nazariy asoslari, filtrlar, konvolyutsiya, g‘ovakliklarni aniqlash kabi ko‘plab muhim masalalar keng yoritilgan[1]. OpenCV kutubxonasi orqali amaliy tasvir tahlilini o‘rganishda Bradski va Kaehlerning Learning OpenCV asari muhim manba bo‘lib xizmat qildi. Unda OpenCV yordamida tasvirni o‘qish, konvertatsiya qilish, filtrdan o‘tkazish va tasvir ustida algoritmik ishlov berish metodlari tushuntirilgan[2]. Shuningdek, Solomon va Breckon tomonidan yozilgan Fundamentals of Digital Image Processing kitobida tasvirlarni segmentatsiya qilish, obyektlarni aniqlash, klassifikatsiya va morfologik ishlov berish kabi ilg‘or usullar haqida batafsil ma’lumotlar keltirilgan[3].

Tadqiqot metodologiyasi. Tasvirni matritsa sifatida qayta ishlash uchun quyidagi yondashuvlardan foydalaniladi. Tasvirni yuklash –

[1] Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing*, 3rd Edition, Pearson Education. **pp. 118–145** – Konvolyutsiya, fazoviy filtrlash va gradient asoslari

[2] Bradski, G., & Kaehler, A. (2008). *Learning OpenCV*, O'Reilly Media. **pp. 45–67** – Tasvirni o‘qish, rang fazolarini o‘zgartirish, blur va boshqa funksiyalar.

[3] Solomon, C., & Breckon, T. (2011). *Fundamentals of Digital Image Processing*, Wiley-Blackwell. **pp. 201–234** – Segmentatsiya, obyekt aniqlash, tasvir chegaralari va morfologik operatsiyalar.

OpenCV kutubxonasidan foydalanib, tasvirni yuklash va uni matritsa ko‘rinishida ifodalash. RGB modelini ishlatish – Har bir piksel uchun RGB qiymatlarini ajratib olish. Ranglarni aniqlash – Ma’lum chegaraviy qiymatlar asosida ranglarni aniqlash va tasniflash. Hisobot chiqarish – Tasvirdagi ranglar soni va ularning ulushini hisoblash. C++ dastur kodida OpenCV kutubxonasi yordamida tasvirni yuklash va uning ranglarini tahlil qilish amalga oshiriladi. OpenCV (Open Source Computer Vision Library) — bu kompyuter ko‘rish va tasvirlarni qayta ishlash uchun mo‘ljallangan bepul kutubxona. U C++, Python va Java kabi tillar bilan ishlaydi. OpenCV tasvir va videolarni qayta ishlash, ob'ektlarni aniqlash, yuzni tanib olish, tasvirni filtrlash va masofani o‘lchash kabi vazifalarni bajarishga imkon beradi. Kutubxona real vaqtda ishlash uchun optimallashtirilgan bo‘lib, ilmiy loyihalar, robototexnika va sun’iy intellekt sohalarida keng qo‘llaniladi.

Tadqiqot natijalari.

C++ dasturlash tilida matritsa orqali tasvir rangini aniqlash uchun tasvirni matritsa sifatida ko‘rib chiqish mumkin. Bu jarayonda tasvir piksellari RGB (Red, Green, Blue) rang qiymatlari sifatida ifodalanadi. Har bir pikselning rangi uning RGB tarkibiga qarab aniqlanadi.

Quyida oddiy misol va OpenCV yordamida tasvir rangini aniqlash haqida ma'lumot keltirilgan.

Oddiy RGB matritsa orqali rangni aniqlash.

Tasvir matritsa sifatida ifodalanadi, va har bir elementning rangi analiz qilinadi.

```
#include <iostream>
using namespace std;
int main()
{
   const int rows = 3, cols = 3;
   // RGB qiymatlar bilan tasvir matritsasi
   int image[rows][cols][3] = {
      {
{255, 0, 0}, {0, 255, 0}, {0, 0, 255}
}, // 1-satr
      {
{255, 255, 0}, {0, 255, 255}, {255, 0, 255}
}, // 2-satr
      {
{128, 128, 128}, {0, 0, 0}, {255, 255, 255}
} // 3-satr
   };
   int redCount = 0, greenCount = 0, blueCount = 0;
   // Ranglarni aniqlash
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         int red = image[i][j][0];
         int green = image[i][j][1];
         int blue = image[i][j][2];
         if (red > green && red > blue) redCount++;      // Qizil
piksel
         else if (green > red && green > blue) greenCount++; //
Yashil piksel
         else if (blue > red && blue > green) blueCount++;   // Ko‘k
piksel
      }
   }
   // Natijalarni chiqarish
   cout << "Qizil piksellar soni: " << redCount << endl;
   cout << "Yashil piksellar soni: " << greenCount << endl;
   cout << "Ko‘k piksellar soni: " << blueCount << endl;
   return 0;
}
```

OpenCV Kutubxonasi yordamida tasvir rangini aniqlash

Agar haqiqiy tasvir fayli bilan ishlash kerak bo‘lsa, OpenCV kutubxonasidan foydalaning. Quyida tasvirni yuklash va ranglarni tahlil qilish misoli keltirilgan.

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main() {
   // Tasvirni yuklash
   Mat image = imread("image.jpg"); // "image.jpg" ni ishlayotgan
papkaga joylashtiring
   if (image.empty()) {
      cout << "Tasvir yuklanmadi!" << endl;
      return -1;
   }
   int redCount = 0, greenCount = 0, blueCount = 0;
   // Har bir pikselni tahlil qilish
   for (int i = 0; i < image.rows; i++)
{
   for (int j = 0; j < image.cols; j++)
{
      Vec3b pixel = image.at<Vec3b>(i, j); // Pikselning RGB
qiymatlari
      int blue = pixel[0];
      int green = pixel[1];
      int red = pixel[2];
      // Ranglarni aniqlash
      if (red > green && red > blue) redCount++;
      else if (green > red && green > blue) greenCount++;
      else if (blue > red && blue > green) blueCount++;
   }
   }
   // Natijalarni chiqarish
   cout << "Qizil rangli piksel soni: " << redCount << endl;
   cout << "Yashil rangli piksel soni: " << greenCount << endl;
   cout << "Ko'k rangli piksel soni: " << blueCount << endl;
   return 0;
}
```

RGB Matritsa: Har bir pikselning qizil (R), yashil (G) va ko‘k (B) qiymatlari 0–255 oralig‘idagi sonlar bilan ifodalanadi.

OpenCV: Tasvirni yuklash va piksellarni boshqarish uchun keng qo‘llaniladigan kutubxona. Tasvir fayllari bilan ishlashni osonlashtiradi.

DLL va Kutubxonalar: OpenCV bilan ishlash uchun tegishli kutubxonalarni o‘rnatish va loyihangizga ulash kerak.

Tasvirdagi ranglarni aniqlashda RGB modelidan foydalanish

Agar tasvirda qizil rang ustunlik qilsa, uning **R** komponenti **G** va **B** komponentlaridan katta bo‘ladi. Xuddi shu tamoyil yashil va ko‘k ranglar uchun ham amal qiladi.

```
if (red > green && red > blue) redCount++;
else if (green > red && green > blue) greenCount++;
else blueCount++;
```

Tasvirdagi ranglarni aniqlashda HSV modelidan foydalanish

HSV (**Hue, Saturation, Value**) modeli yordamida tasvir ranglarini aniqroq aniqlash mumkin. HSV modeli rang shkalasi, to‘yinganlik va yorqinlikni hisobga oladi.

```
Mat hsv;
cvtColor(img, hsv, COLOR_BGR2HSV);
for (int i = 0; i < hsv.rows; i++)
{
   for (int j = 0; j < hsv.cols; j++)
{
      Vec3b pixel = hsv.at<Vec3b>(i, j);
      int h = pixel[0];  // Hue
      int s = pixel[1];  // Saturation
      int v = pixel[2];  // Value
      if (h >= 0 && h <= 10) redCount++;
      else if (h >= 35 && h <= 85) greenCount++;
      else if (h >= 100 && h <= 140) blueCount++;
   }
}
```

Quyidagi sohalarda keng qo‘llaniladi

1. **Yuzni aniqlash** – teri rangini tanib olish uchun
2. **Tibbiyot** – tahlil natijalari bo‘yicha diagnostika qilish
3. **Sanoat** – mahsulot sifati nazorati uchun

Tasvirni Tiniqlashtirish (Image Sharpening) C++ va OpenCV orqali Tasvirni qayta ishlashda tiniqlikni oshirish muhim vazifalardan biridir. Bu, ayniqsa, tibbiyot, sun’iy intellekt, avtomatlashtirilgan kuzatuv

tizimlari va sanoat monitoringi kabi sohalarda katta ahamiyatga ega. Ushbu maqolada C++ dasturlash tilida OpenCV kutubxonasidan foydalangan holda tasvirni tiniqlashtirish (sharpening) usullari ko‘rib chiqiladi. Sobel filtrlari, Laplasian filtri va Unsharp Masking texnikalaridan foydalanish orqali tasvirning tiniqligini oshirish algoritmlari taqdim etiladi.

Tasvirni tiniqlashtirish uchun quyidagi usullar qo‘llaniladi:

Sobel filtri – Gradient hisoblash orqali tasvir chetlarini aniqroq chiqarish.

Laplasian filtri – Ikkinchi tartibli hosila orqali tiniqlikni oshirish.

Unsharp Masking – Past chastotali filtrni olib tashlash orqali tasvirni tiniqlashtirish.

1. Sobel Filtri

Sobel filtri tasvirning chetlarini aniqroq ko‘rsatishga yordam beradi. C++ va OpenCV yordamida uni quyidagicha qo‘llash mumkin:

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main()
 {
   Mat img = imread("image.jpg", IMREAD_GRAYSCALE);

   if (img.empty())
{
     cout << "Tasvir yuklanmadi!" << endl;
     return -1;
   }
   Mat grad_x, grad_y, sobel_img;
   Sobel(img, grad_x, CV_16S, 1, 0, 3);
   Sobel(img, grad_y, CV_16S, 0, 1, 3);
   convertScaleAbs(grad_x, grad_x);
   convertScaleAbs(grad_y, grad_y);
   addWeighted(grad_x, 0.5, grad_y, 0.5, 0, sobel_img);
   imshow("Asl Tasvir", img);
   imshow("Sobel Tiniqlashtirilgan Tasvir", sobel_img);
   waitKey(0);
   return 0;
}
```

Natija

Sobel filtri chekkalarni aniqroq ko‘rsatadi va tasvirning detallarini ajratib beradi. Biroq, ba’zi hollarda shovqinni ham oshirib yuborishi mumkin.

2. Laplasian Filtri

Laplasian operatori tasvirning ikkinchi tartibli hosilalarini hisoblab, tiniqlikni oshirishga yordam beradi.

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main()
{
   Mat img = imread("image.jpg", IMREAD_GRAYSCALE);

   if (img.empty())
{
     cout << "Tasvir yuklanmadi!" << endl;
     return -1;
   }

   Mat laplacian_img, result;
   Laplacian(img, laplacian_img, CV_16S, 3);
   convertScaleAbs(laplacian_img, laplacian_img);
     addWeighted(img, 1.5, laplacian_img, -0.5, 0, result);
   imshow("Asl Tasvir", img);
   imshow("Laplasian Tiniqlashtirilgan Tasvir", result);
   waitKey(0);
   return 0;
}
```

Natija

Laplasian filtri tasvir chetlarini keskinroq chiqaradi va tiniqlikni oshiradi.

3. Unsharp Masking

Unsharp Masking – tasvirni tiniqlashtirishning eng samarali usullaridan biri. Bu usulda tasvirdan past chastotali filtri olib tashlanib, kontrast oshiriladi.

```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main()
 {
   Mat img = imread("image.jpg", IMREAD_GRAYSCALE);
   if (img.empty())
{
     cout << "Tasvir yuklanmadi!" << endl;
     return -1;
   }
   Mat blurred, sharp_img;
   GaussianBlur(img, blurred, Size(9, 9), 10);
   addWeighted(img, 1.5, blurred, -0.5, 0, sharp_img);
   imshow("Asl Tasvir", img);
   imshow("Unsharp Masking Tiniqlashtirilgan Tasvir",
sharp_img);
   waitKey(0);
   return 0;
}
```

Unsharp Masking tasvirni tiniqlashtirishda juda yaxshi natija beradi va shovqinni kamaytiradi.

Natijalar va tahlili

Usul	Afzalliklari	Kamchiliklari
Sobel	Chekkalarni yaxshiroq aniqlaydi	Shovqinlarni oshirishi mumkin
Laplasian	Tez ishlaydi va oddiy	Ba’zan ortiqcha tiniqlik hosil qiladi
Unsharp Masking	Eng yaxshi tiniqlik natijasi	Hisoblash jihatdan nisbatan qiyinroq

C++ va OpenCV yordamida tasvirni tiniqlashtirish usullari ko‘rib chiqildi. Sobel, Laplasian va Unsharp Masking usullari yordamida tasvirning sifatini yaxshilash imkoniyati aniqlandi.

Sobel filtri chekkalarni yaxshiroq chiqarish uchun mos keladi.

Laplasian filtri chekka va konturlarni tiniqlashtirishda foydalidir.

Unsharp Masking eng yaxshi tiniqlik natijalarini beradi.

Ushbu texnikalarni sun’iy intellekt algoritmlari bilan birlashtirib, avtomatik tiniqlik optimallashtirish tizimlarini yaratish mumkin.

Super Resolution (Sun’iy intellect asosida). Bu usul mashinani o‘rganish (Machine Learning) yoki sun’iy intellekt yordamida past sifatli tasvirlarni yuqori sifatli qilib qayta ishlaydi. Eng mashhur modellari ESRGAN (Enhanced Super-Resolution GAN), Waifu2x, SRCNN (Super-Resolution Convolutional Neural Network).

Afzalliklari:	*Kamchiliklari:*
Detallarni yaxshiroq tiklaydi. Shovqin kamayadi va tabiiy ko‘rinadi. Past sifatli rasmlarni yuqori sifatga olib kelish imkoniyati mavjud.	Hisoblash resurslari ko‘proq talab etadi. Har doim ham 100% aniq natija bermasligi mumkin. Ba’zi joylarda noto‘g‘ri detallar tiklanishi mumkin.

Taqqoslash jadvali

Xususiyat	Filtrlash usuli	Sun'iy intellekt (Super Resolution)
Ishlash tezligi	Tez	Sekinroq (katta hisoblash talab etadi)
Detallarni tiklash	Chegaralarni tiniqlashtiradi	Yo'qolgan detallarni qayta yaratadi
Shovqin (Noise)	Shovqinni kuchaytirishi mumkin	Shovqinni kamaytiradi
Qo'llash sohalari	Suratlarni yaxshilash, fotoshop	Video tiniqlashtirish, past sifatli rasmlarni tiklash
Asboblar	Photoshop, OpenCV	ESRGAN, Waifu2x, SRCNN

Agar tasvirda faqat biroz tiniqlik kerak bo'lsa, filtrlash usuli yaxshi tanlov. Agar sifati juda past bo'lsa va detallarni tiklash kerak bo'lsa, Sun'iy intellekt usuli samarali bo'ladi.

Super Resolution texnologiyasi past sifatli tasvirlarni yuqori sifatga olib chiqish uchun ishlatiladi. Ko'pincha konvolyutsion neyron tarmoqlari (CNN) va generativ adversarial tarmoqlar (GAN) yordamida amalga oshiriladi.

Eng mashhur modellar:

- **SRCNN (Super-Resolution Convolutional Neural Network)**
- **ESRGAN (Enhanced Super-Resolution GAN)**
- **FSRCNN (Fast Super-Resolution CNN)**
- **Waifu2x**

ESRGAN asosida Super Resolution dasturini keltirib o'tman. Bu model yuqori sifatli tasvirlarni yaratishga yordam beradi.

1. Talab qilinadigan kutubxonalarni o'rnatish

Python-da Super Resolution uchun quyidagi kutubxonalar kerak bo'ladi:

```
pip install torch torchvision numpy opencv-python matplotlib
```

2. Super Resolution uchun dastur kod

```
import cv2
import torch
import numpy as np
import torchvision.transforms as transforms
from torchvision.utils import save_image
from PIL import Image
import matplotlib.pyplot as plt

# ESRGAN modelini yuklash
model_path = "ESRGAN.pth"  # Model fayli yuklangan bo'lishi kerak
model = torch.load(model_path, map_location=torch.device("cpu"))
model.eval()

# Tasvirni yuklash va tayyorlash
def preprocess_image(image_path):
    image = Image.open(image_path).convert("RGB")
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
    return transform(image).unsqueeze(0)

# Tasvirni Super Resolution orqali yaxshilash
def super_resolve(image_path, output_path):
    input_image = preprocess_image(image_path)
    with torch.no_grad():
        sr_image = model(input_image)
    sr_image = sr_image.squeeze(0).permute(1, 2, 0).numpy()
    sr_image = ((sr_image + 1) / 2.0 * 255).astype(np.uint8)  # Normallashtirish
    cv2.imwrite(output_path, cv2.cvtColor(sr_image, cv2.COLOR_RGB2BGR))

# Natijani tekshirish
input_image_path = "low_quality.jpg"
output_image_path = "high_quality.jpg"

super_resolve(input_image_path, output_image_path)

# Natijani ko'rsatish
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].imshow(cv2.imread(input_image_path)[:, :, ::-1])
ax[0].set_title("Past sifatli rasm")
ax[1].imshow(cv2.imread(output_image_path)[:, :, ::-1])
ax[1].set_title("Super Resolution natijasi")
plt.show()
```

3. Dastur kodining ishlashi quyidagicha tavsiflanadi.

1. **Tasvirni yuklaydi** va kerakli shaklga o'tkazadi.
2. **ESRGAN modelidan foydalanib**, yuqori sifatli tasvirni yaratadi.
3. **Natijani saqlaydi va vizual ko'rsatadi**.

4. Qo'shimcha ma'lumotlar

Model fayli: Agar siz ESRGAN modelidan foydalanmoqchi bo'lsangiz, oldindan tayyorlangan "**ESRGAN.pth"** modelini yuklab olishingiz kerak. Modelni ushbu havoladan https://github.com/xinntao/ESRGAN yuklab olishingiz mumkin.

Dastur real vaqtda ishlaydi: Model CUDA (GPU) bilan ham ishlashi mumkin, bu ish tezligini oshiradi.

Kirish tasviri: Har qanday past sifatli tasvirni yuklab, natijani solishtirishingiz mumkin.

Xulosa qilib aytganda, ushbu kod past sifatli tasvirlarni yuqori sifatga o'tkazib beradi. Agar real vaqt rejimida video yoki boshqa ilovalar uchun ishlatmoqchi bo'linsa, kodni o'zgartirish mumkin.

Super Resolution texnologiyasining asosiy afzalligi – tasvirni realistik va aniq holatga keltirish imkoniyati. Biroq, bu jarayon resurs talab qiladi va GPU yoki maxsus model talab etadi. Shu bilan birga, an'anaviy C++ va OpenCV usullari yordamida filtrlar orqali tasvirni tiniqlashtirishning ham samarali ekani aniqlandi. Ushbu texnologiyalar kelajakda yanada rivojlanib, tezkor va sifatli natijalar taqdim etishi kutilmoqda.

Foydalanilgan adabiyotlar ro'yxati.

1. Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing*. Pearson.
2. Oppenheim, A. V., & Schafer, R. W. (2010). *Discrete-Time Signal Processing*. Prentice Hall.
3. Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media.
4. Mansurovich, B. M., & Ogli, Y. M. D. (2022). PHP dasturlash tili va uning imkoniyatlari. *Ta'lim fidoyilari*, *18*(5), 77-80.
5. Aripov, M. M., Normatov, R. N., Siddikov, I. M., & Oripova, U. (2020). Fundamentals of creating the algebra science and algorithms. *Solid state technology*, *63*(5), 6103-6111.
6. Ботиров, М. М. (2023). Мировой опыт развития профессиональных компетенций студентов на основе цифровых технологий. *Conferencea*, 18-22.