MTC True Tech

ЛЕКЦИЯ 2

Иван Копылов

CV Engineer

МТС x НИУ ВШЭ

Введение

Под изображением в контексте компьютерного зрения чаще всего понимают двумерную функцию I(x, y), где x и y — координаты в пространстве пикселей, а значение I(x, y) — интенсивность (яркость или цвет) в данной точке. В реальном мире мы имеем дело с непрерывным распределением света, но цифровые системы работают с конечными наборами дискретных точек (пикселей) и ограниченным числом уровней (квантование)

Компьютеры хранят изображения в виде растровых данных: это сетка из M × N точек, где каждая точка (пиксель) имеет одно или несколько числовых значений. Например:

Дискретизация и квантование: переход к цифровому формату

Любое реальное изображение в природе — это непрерывное распределение интенсивности света по поверхности. Чтобы компьютер мог обрабатывать снимок, выполняются два ключевых шага:

Дискретизация по пространству (sampling)

Мы разбиваем непрерывное поле на дискретную сетку пикселей. Чем гуще сетка, тем выше разрешение и детальность

Квантование по интенсивности (quantization)

Интенсивность (или цвет) пикселя округляется до ближайшего значения из ограниченного набора уровней. Например, при 8-битном представлении для одного канала есть 2^8 = 256 уровней

Основные характеристики изображений

Разрешение

Разрешение изображения определяет, сколько пикселей по горизонтали (width) и вертикали (height). Примеры: HD: 1280×720, Full HD: 1920×1080, 4K: 3840×2160

Глубина цвета

Глубина цвета (color depth) — это количество бит, отведённых на хранение значения интенсивности/цвета. 8 бит на канал (общее 24 бита для RGB): 256 уровней на канал, всего 256^3 ≈ 16.7 млн цветов

Форматы хранения

- JPEG: сжатие с потерями.
- PNG: сжатие без потерь, поддержка прозрачности.
- BMP: формат Windows, без сжатия.
- TIFF: может хранить много бит на канал.
- RAW: содержит несжатые данные с матрицы камеры.

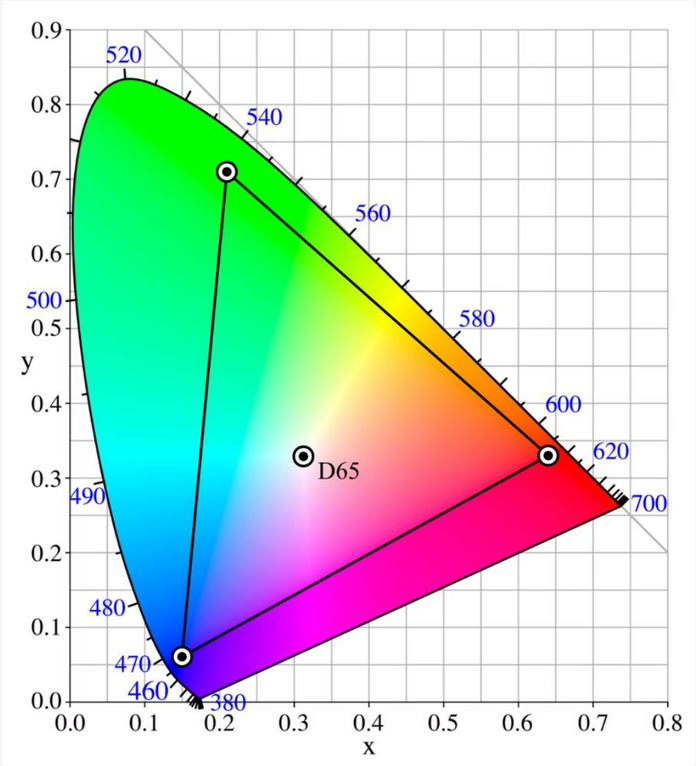
Математическая модель изображения

Хотя на практике мы имеем дело с пикселями, полезно понимать непрерывную (аналитическую) модель

Пусть I(x,y) — интенсивность в точке (x,y). Если речь о цветном изображении, у нас три такие функции:

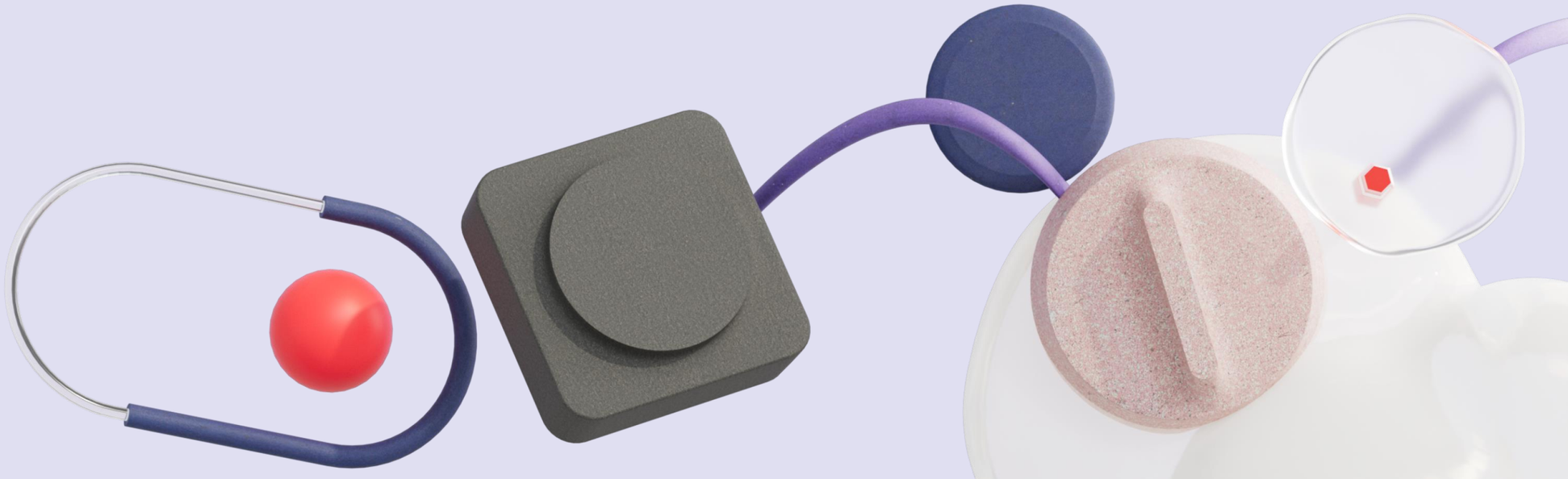
- I_R(x,y) — интенсивность красного,
- I_G(x,y) — зелёного,
- I_B(x,y) — синего.

Для удобства обработки часто представляют эти три компоненты как 3-мерный вектор I(x,y) = (I_R, I_G, I_B). Тогда операции можно записывать в векторной форме



ЦВЕТОВЫЕ ПРОСТРАНСТВА: НАЗНАЧЕНИЕ, ОСОБЕННОСТИ, ПЛЮСЫ И МИНУСЫ

Разберём более детально популярные цветовые модели, чтобы понимать, когда и зачем их использовать при анализе изображений

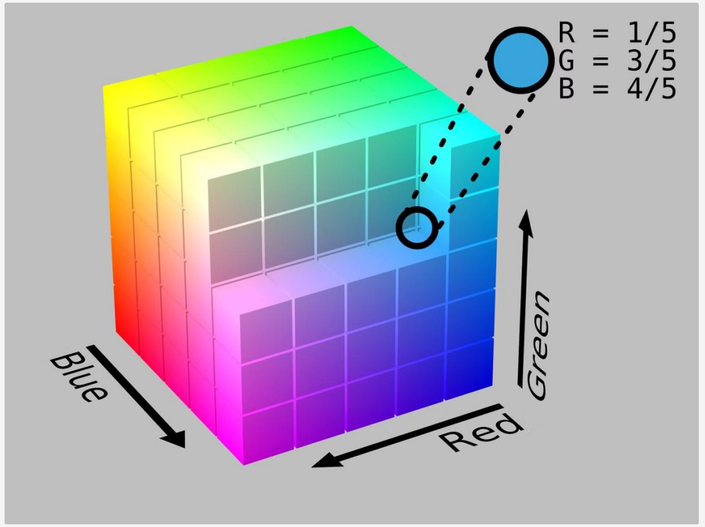


МТС x НИУ ВШЭ

RGB (Red, Green, Blue)

Как получается цвет

Аддитивное смешение красного, зелёного, синего излучения. Так работают экраны, мониторы, телевизоры



Удобство

Большинство исходных изображений в компьютерном зрении считываются в BGR/RGB. Для элементарных операций (отображение, простая фильтрация) это вполне хорошо

Сложности

Если хотим сегментировать объекты по цвету, изменение освещения существенно влияет на (R,G,B)-координаты. В RGB часто яркость смешана с цветом, поэтому выделить тени и блики может быть трудно

Grayscale (градации серого)

Сущность: I_gray(x,y) = 0.299*R + 0.587*G + 0.114*B.

Использование

Упрощает многие задачи (выделение контуров, анализ текстур), если цветовая информация не нужна. Уменьшение объёма данных в 3 раза (один канал вместо трёх)

В ряде задач (OCR, бинарная сегментация) достаточно работать в «сером» формате, что упрощает алгоритмы и ускоряет вычисления

Пример кода

import cv2 image = cv2.imread("example.jpg") # BGR по умолчанию gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

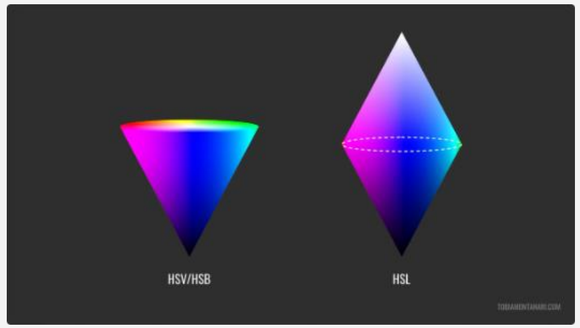
HSV (Hue, Saturation, Value)

Описание

Hue (оттенок, 0…360° или 0…180 в OpenCV),
Saturation (насыщенность), Value (яркость)

Преимущества

Можно искать объекты по цвету (Hue) независимо
от освещения (Value). Удобно создавать цветовые фильтры



Недостатки

При низкой насыщенности (S→0) оттенок становится
неопределённым. Hue циклический, то есть 0° и 360° —
это один и тот же цвет (красный)

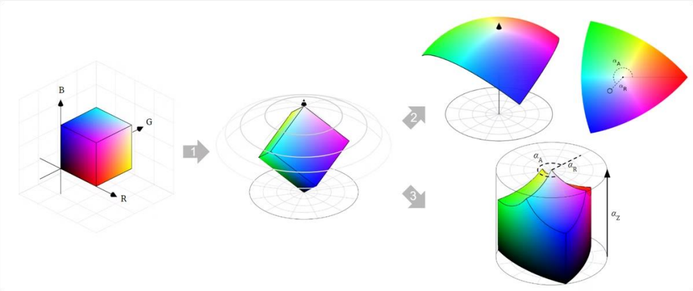
LAB (Lightness, A, B)

Как устроено

L — ось яркости (0 — чёрное, 100 — белое), A —
от зелёного к красному, B — от синего к жёлтому.

Близость к человеческому восприятию

Изменения в L влияют на яркость, а каналы
A, B описывают «цвет» без учёта освещения



Применение

Коррекция цвета: можно повысить контраст только
в канале L, не трогая цветовую составляющую. Сегментация:
иногда выделение объекта проще
в LAB, если он отличается по «A» или «B» от фона

YCbCr (яркость и цветность)

Описание

Y — яркость (luma), Cb, Cr — компоненты цветности (chroma)

Где используется

Стандарты телевидения (PAL, NTSC), видеокодеки (MPEG). JPEG также использует похожую схему YUV (субдискретизация цветовых каналов)

Преимущество

Можно сжимать Cb, Cr сильнее (человек меньше замечает потери в цвете, чем в яркости)

Преобразования между цветовыми пространствами в OpenCV

В практических задачах компьютерного зрения часто приходится переключаться из одного цветового пространства в другое. Например, мы можем:

- Считать изображение в BGR (стандарт OpenCV)
- Конвертировать в HSV, чтобы сегментировать по оттенку
- Конвертировать результат в Grayscale для выделения краёв

RGB → Grayscale

```
import cv2

image = cv2.imread("example.jpg")
# BGR формат

gray_image = cv2.cvtColor
(image, cv2.COLOR_BGR2GRAY)
```

BGR → HSV

```
hsv_image = cv2.cvtColor(image,
cv2.COLOR_BGR2HSV)
```

BGR → LAB

```
lab_image = cv2.cvtColor(image,
cv2.COLOR_BGR2LAB)
```

Гистограммы и их роль в анализе изображений

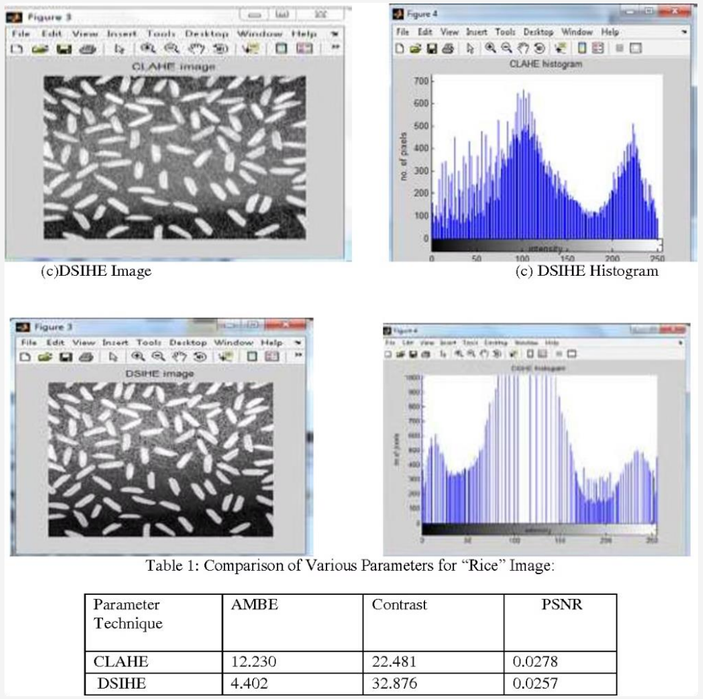
Гистограмма интенсивностей (или цветовых каналов) показывает, сколько пикселей имеют ту или иную величину яркости (или цветового компонента). Она даёт представление о распределении тонов в изображении

Гистограмма для grayscale

Ось X — интенсивность 0…255, ось Y — число пикселей.

Гистограмма для цветного изображения

Строится отдельно для R, G, B (либо для H, S, V и т. д.).



Выравнивание гистограммы: улучшение контраста

Выравнивание гистограммы (histogram equalization) перераспределяет интенсивности так, чтобы гистограмма стала более плоской и широкого диапазона. Это позволяет улучшить контраст, особенно если исходное изображение «сжат» в узком диапазоне яркостей

Глобальное выравнивание (EqualizeHist)

```
import cv2

image = cv2.imread("dark_image.jpg", cv2.IMREAD_GRAYSCALE) equalized = cv2.equalizeHist(image)
```

Локальное выравнивание (CLAHE)

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8)) clahe_result = clahe.apply(image)
```

Методы сглаживания и подавления шума

Среднее сглаживание (Box Blur)

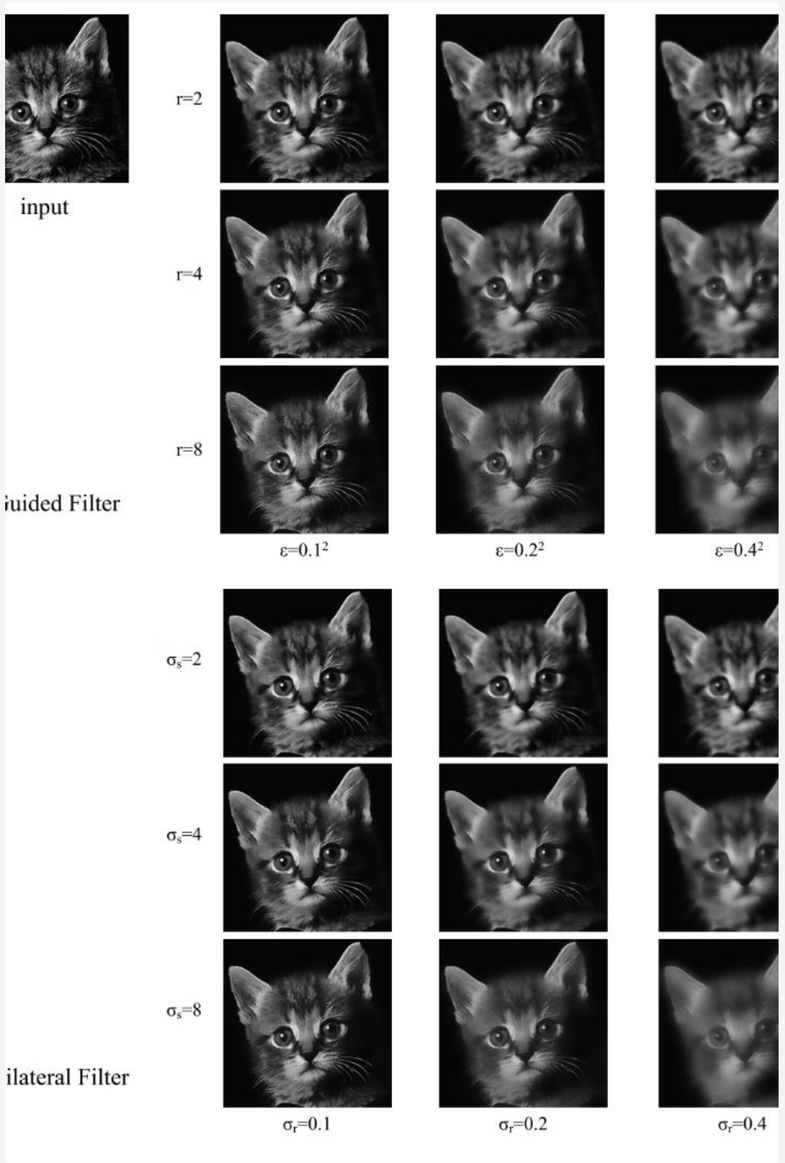
import cv2 blurred = cv2.blur(image, (5, 5))

Гауссово сглаживание (Gaussian Blur)

gaussian = cv2.GaussianBlur(image, (5, 5), 0)

Медианный фильтр (Median Blur)

median = cv2.medianBlur(image, 5)



Увеличение резкости и нормализация яркости/контраста

Фильтр повышения резкости (Sharpening)

import cv2 import numpy as np

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

sharpened = cv2.filter2D(image, -1, kernel)

Нормализация

mean_val = np.mean(image) std_val = np.std(image) standardized = (image - mean_val) / std_val

normalized = cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX)

Пороговая сегментация и её вариации

Фиксированный порог (Global Threshold)

```
import cv2 image = cv2.imread("example.jpg", cv2.IMREAD_GRAYSCALE)
ret, binary = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
```

Адаптивная пороговая обработка (Adaptive Threshold)

```
adaptive = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE
_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
```

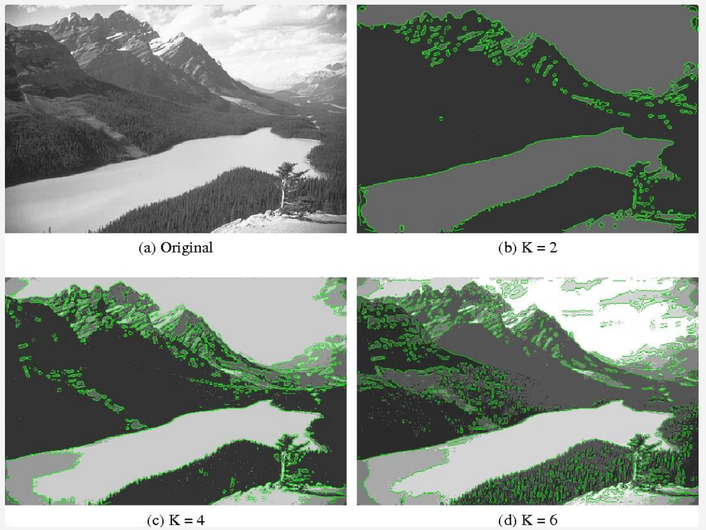
Порог по Оцу (Otsu's Thresholding)

```
ret, otsu_binary = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

Кластеризация K-means: разбиение на группы по цвету или интенсивности

K-means — метод, который пытается разбить множество точек на k кластеров, минимизируя сумму квадратов расстояний до центров. В контексте изображений:

1. Мы рассматриваем каждый пиксель как вектор (R,G,B) либо (L,A,B).
2. Хотим поделить все пиксели на k групп (например, k=2 — фон и объект, k=3 — небо, земля, деревья и т. п.).



Морфологические операции

Эрозия (Erosion)

import cv2 import numpy as np kernel = np.ones((3, 3), np.uint8) eroded = cv2.erode(binary_image, kernel, iterations=1)

Дилатация (Dilation)

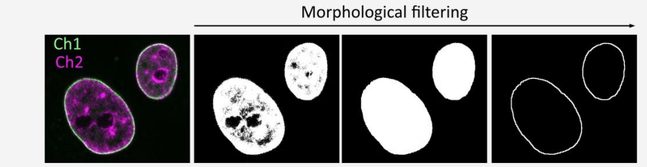
dilated = cv2.dilate(binary_image, kernel, iterations=1)

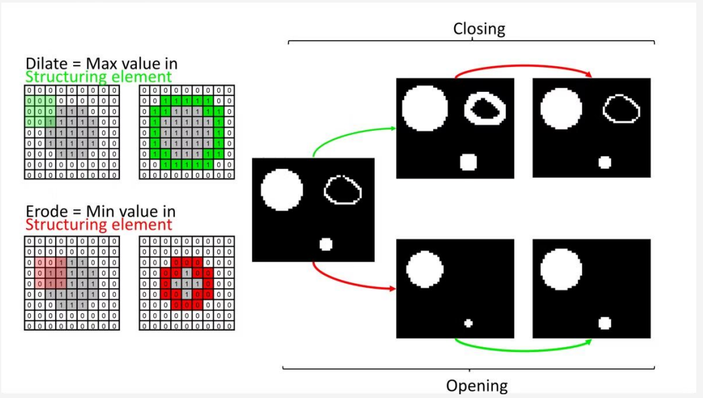
Открытие (Opening)

opened = cv2.morphologyEx(binary_image, cv2.MORPH_OPEN, kernel)

Закрытие (Closing)

closed = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)





Детекторы краёв (Собель, Лапласиан, Канни)

Оператор Собеля

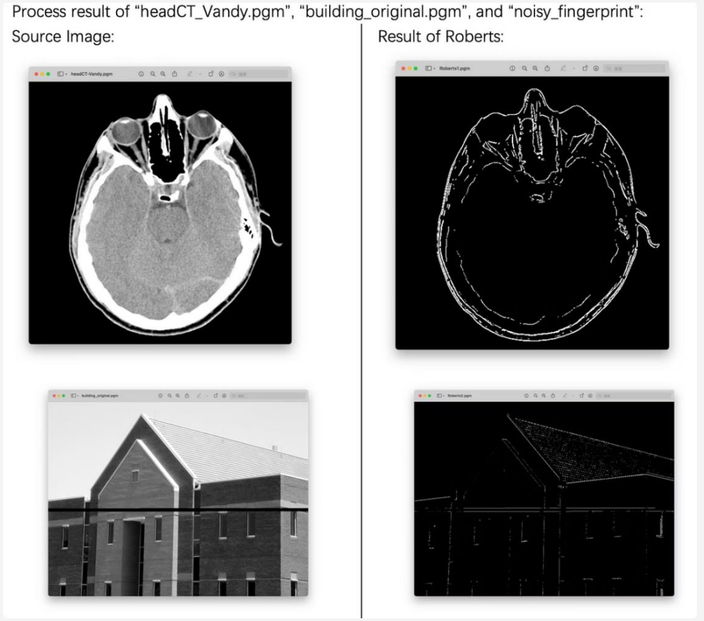
import cv2 sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3) sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3) magnitude = cv2.magnitude(sobelx, sobely)

Лапласиан (Laplacian)

lap = cv2.Laplacian(image, cv2.CV_64F)

Алгоритм Канни (Canny Edge Detector)

edges = cv2.Canny(image, 100, 200)



Ключевые точки и дескрипторы

Детекторы углов (Harris, Shi-Tomasi)

corners = cv2.cornerHarris(gray, 2, 3, 0.04) corners = cv2.goodFeaturesToTrack(gray, maxCorners=100, qualityLevel=0.01, minDistance=10)

SIFT (Scale-Invariant Feature Transform)

sift = cv2.SIFT_create() kp, des = sift.detectAndCompute(gray, None)

ORB (Oriented FAST and Rotated BRIEF)

orb = cv2.ORB_create() kp, des = orb.detectAndCompute(gray, None)

Сопоставление дескрипторов и построение панорам

Brute-Force Matcher

```
import cv2

 bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True) matches = bf.match(descriptors1, descriptors2) matches = sorted(matches, key=lambda x: x.distance)
```

FLANN

```
import numpy as np

FLANN_INDEX_KDTREE = 1

index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)

search_params = dict(checks=50) flann = cv2.FlannBasedMatcher(index_params, search_params) matches = flann.knnMatch(descriptors1, descriptors2, k=2)
```

Методы сегментации изображений: разбор (Watershed, GrabCut)

Алгоритм Watershed

markers = np.zeros_like(gray_image, dtype=np.int32) # заполнить markers начальными метками объектов и фона cv2.watershed(color_image, markers)

GrabCut

mask = np.zeros(image.shape[:2], np.uint8) bgdModel = np.zeros((1, 65), np.float64) fgdModel = np.zeros((1, 65), np.float64) rect = (50, 50, 200, 200) # Прямоугольник, где сидит объект cv2.grabCut(image, mask, rect, bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_RECT)

Методы улучшения изображения: билатеральный фильтр, суперрезолюшн

Билатеральный фильтр (Bilateral Filter)

smoothed = cv2.bilateralFilter(image, d=9, sigmaColor=75, sigmaSpace=75)

Суперрезолюшн (Super-Resolution)

import cv2 sr = cv2.dnn_superres.DnnSuperResImpl_create() sr.readModel("ESPCN_x4.pb")
sr.setModel("espcn", 4) high_res = sr.upsample(low_res)

Подготовка изображений к обучению нейросетей

Изменение размера (Resize)

resized = cv2.resize(image, (224, 224))

Нормализация

import numpy as np image = np.float32(image) / 255.0 mean = [0.485, 0.456, 0.406] std = [0.229, 0.224, 0.225] for c in range(3): image[:, :, c] = (image[:, :, c] - mean[c]) / std[c]

Аугментация

import random def random_flip(image): if random.random() > 0.5: image = cv2.flip(image, 1) # flip horizontally return image

Метрики оценки качества обработки и примеры применения (PSNR, SSIM)

PSNR (Peak Signal-to-Noise Ratio)

def psnr(img1, img2): mse = np.mean((img1 - img2) ** 2) if mse == 0:
return float('inf') return 10 * np.log10((255**2)/mse)

SSIM (Structural Similarity Index)

from skimage.metrics import structural_similarity as ssim score, diff
= ssim(img1, img2, full=True)

Дополнительные примеры практического применения

Промышленность

Контроль качества: На конвейере камера делает снимки деталей. Система выявляет дефекты (трещины, сколы) путём анализа текстур, сравнения с эталоном или сегментации

Медицина

Анализ снимков (рентген, КТ, МРТ): Улучшение контраста, шумоподавление, сегментация органов или опухолей. Выделяют ключевые области, подсчитывают площадь, объём

Космические исследования

Спутниковые снимки: Сегментация ландшафта (водоёмы, леса, поля), детекция облаков, снежного покрова

Резюме лекции и перспективы использования данных методов

В данной лекции рассмотрены ключевые техники обработки изображений, необходимые для компьютерного зрения:

Основы

Представление изображений, цветовые пространства, гистограммы и их выравнивание

Обработка

Сглаживание шума, увеличение резкости, сегментация, морфологические операции

Анализ

Детекторы краёв, ключевые точки и дескрипторы, сопоставление и построение панорам

Продвинутые методы

Билатеральный фильтр, суперрезолюшн, подготовка к нейросетям, метрики оценки качества.

Перспективы: С повышением разрешения камер и появлением новых сенсоров методы анализа изображений постоянно совершенствуются. Растет спрос на автоматическую сегментацию в медицине и надежную детекцию в реальном времени для автономных систем

MTC True Tech

СПАСИБО ЗА ВНИМАНИЕ

Иван Копылов

CV Engineer

МТС x НИУ ВШЭ