

Лекция 1. Природа изображения

Цель: понять изображение как сигнал — от света и непрерывного поля к дискретной матрице, частотам, шумам, цветовым пространствам и форматам.

Ключевые темы: дискретизация, квантование, свёртка, алиасинг, яркость/контраст/гистограммы, цвет (RGB/HSV/Lab/YCbCr), форматы (BMP/PNG/JPEG), фильтры и спектр.

Раздел 1. Изображение как сигнал

1.1. Физическая природа изображения как светового поля

Изображение — регистрация поля освещённости: интеграл по спектру с учётом чувствительности сенсора $S(\lambda)$.

$$I(x, y) = \int_{\lambda_{\min}}^{\lambda_{\max}} E(x, y, \lambda) S(\lambda) d\lambda$$

Камера усредняет по времени и дискретизирует по пространству, превращая поле в матрицу чисел.

1.2. От непрерывного к дискретному

Пространственное сэмплирование на сенсоре $M \times N$: усреднение яркости по ячейке размера $\Delta x, \Delta y$.

$$I[m, n] = \frac{1}{\Delta x \Delta y} \int_{x_m - \Delta x/2}^{x_m + \Delta x/2} \int_{y_n - \Delta y/2}^{y_n + \Delta y/2} I(x, y) dx dy$$

При превышении половины частоты дискретизации возникает **алиасинг** (муар, ложные структуры).

1.3. Квантование и динамический диапазон

После сэмплирования выполняется квантование на $L=2^n$ уровняй.

$$I[m, n] \in [0, L - 1], \quad L = 2^n$$

8 бит \rightarrow 256 уровняй. Большая битность расширяет динамический диапазон и снижает квантовый шум, делая тона плавнее.

1.4. Изображение как дискретный сигнал

Монохром: двумерная дискретная функция целочисленных аргументов.

$$I[m, n] : \mathbb{Z}^2 \rightarrow [0, L - 1]$$

Операции (фильтрация, дифференцирование, восстановление) — линейные/нелинейные преобразования дискретных функций.

1.5. Пространственные частоты и спектр

Частоты: медленные изменения — низкие, быстрые — высокие. 2D-ДПФ раскладывает на синусоиды.

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y) e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

Центр спектра — низкие частоты, края — высокие; основа сглаживания и выделения контуров.

1.6. Свёртка – фундамент фильтрации

Локальная операция с ядром $K(u,v)$ реализует усреднение/усиление частот.

$$(I * K)(x, y) = \sum_{u=-m}^m \sum_{v=-n}^n I(x - u, y - v) K(u, v)$$

Оператор Лапласа усиливает контуры (дискретная вторая производная яркости).

```
kernel = np.array([[1,1,1],[1,-8,1],[1,1,1]], np.float32)
img = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
edges = cv2.filter2D(img, -1, kernel)
```

1.7. Алиасинг и антиалиасинг

Условие Найквиста в 2D: частоты сигнала ниже предельных по шагам $\Delta x, \Delta y$.

$$\omega_x < \pi/\Delta x, \quad \omega_y < \pi/\Delta y$$

Для подавления высоких частот перед сэмплированием применяют гауссово размытие.

```
blur = cv2.GaussianBlur(img, (5, 5), 1.2)
```

1.8. Связь областей: свёртка и умножение

Теорема о свёртке: фильтрация в пространстве равна умножению спектров.

$$\mathcal{F}\{I * K\}(u, v) = F_I(u, v) \cdot F_K(u, v)$$

Большие фильтры часто быстрее применять в частотной области; важно учитывать граничные эффекты.

1.9. Модель шума

Сенсор вносит аддитивный гауссов шум нулевого среднего.

$$I_{\text{obs}}(x, y) = I_{\text{true}}(x, y) + \eta(x, y), \quad \eta \sim \mathcal{N}(0, \sigma^2)$$

Шум усиливает высокие частоты; сглаживающие фильтры подавляют шум, сохраняя структуру.

1.10. Итог: сигнал и восприятие

Изображение — дискретный 2D-сигнал с параметрами: разрешение, битность, частоты, шум.

Анализ опирается на линейность, спектры, фильтрацию и устойчивость; человек воспринимает нелинейно.

Раздел 2. Яркость, контраст и гистограммы

2.1. Понятие яркости и восприятия

Глаз реагирует нелинейно; удобна логарифмическая шкала Вебера–Фехнера.

$$B = k \log(1 + \alpha E)$$

Линейное масштабирование интенсивности не даёт визуально линейного эффекта, особенно в тенях.

2.2. Контраст как относительная мера

Контраст Мишельсона для двух областей I_{\max} и I_{\min} :

$$C = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$

Статистическая мера по всему изображению — стандартное отклонение яркости.

$$C_{\text{std}} = \sqrt{\frac{1}{MN} \sum_{m,n} (I[m, n] - \bar{I})^2}$$

2.3. Линейные преобразования яркости

Изменяем контраст и смещение: α — масштаб диапазона, β — яркость.

$$I'(x, y) = \alpha I(x, y) + \beta$$

```
img = cv2.imread("city.jpg", cv2.IMREAD_GRAYSCALE)
alpha, beta = 1.2, 20
adjusted = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)
```

Автообрезка предотвращает клиппинг вне 0–255.

2.4. Гистограмма яркости

Нормированная гистограмма $p(i)$ — оценка плотности вероятности яркости.

$$p(i) = \frac{h(i)}{MN}, \quad \sum_{i=0}^{L-1} p(i) = 1$$

```
hist = cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist); plt.title("Гистограмма яркости")
```

Положение и ширина гистограммы отражают экспозицию и контраст.

2.5. Растворение и нормализация

Линейно растворяем диапазон [I_{\min} , I_{\max}] к [0,255] для усиления контраста.

$$I'(x, y) = \frac{I(x, y) - I_{\min}}{I_{\max} - I_{\min}} \cdot 255$$

Улучшает различимость, но может усилить шум при узком исходном диапазоне.

2.6. Эквализация гистограммы

Делаем распределение яркости равномернее через CDF.

$$c(i) = \sum_{j=0}^i p(j), \quad I'(x, y) = (L - 1) c(I(x, y))$$

```
equalized = cv2.equalizeHist(img)
```

Полезна при неравномерном свете; при сильной неоднородности — CLAHE.

2.7. Локальная контрастность (CLAHE)

Эквализация в блоках с ограничением усиления контраста (clipLimit).

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileSize=(8,8))
local_eq = clahe.apply(img)
```

Хорошо для медицинских/спутниковых снимков: выравнивает детали, сдерживает шум.

2.8. Освещённость vs отражение

Модель мультипликативна: освещённость $L(x,y)$ и отражение $R(x,y)$.

$$I(x, y) = L(x, y) R(x, y)$$

Логарифм переводит в сумму; гомоморфная фильтрация компенсирует неравномерность света.

2.9. Комплексная цепочка обработки

Нормализация → глобальная эквализация → CLAHE для извлечения деталей.

```
img = cv2.imread("dark_scene.jpg", cv2.IMREAD_GRAYSCALE)
norm = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
equalized = cv2.equalizeHist(norm)
clahe = cv2.createCLAHE(3.0,(8,8))
result = clahe.apply(equalized)
```

Оценивайте эффект визуально и метриками (например, C_std).

2.10. Практический вывод

Яркость, контраст и гистограммы — базовые инструменты предобработки перед сегментацией, контурной и нейросетевой обработкой.

Нормализация распределений повышает устойчивость алгоритмов.

Раздел 3. Цвет и цветовые пространства

Цвет — восприятие, обусловленное спектром и трихроматической системой (колбочки S, M, L). Камеры имитируют три канала.

$$\mathbf{I}(x, y) = [I_R(x, y), I_G(x, y), I_B(x, y)]^T$$

RGB — инженерная аппроксимация восприятия.

3.1. Физиология и физика цвета

Три канала (S,M,L) образуют триадную систему; сенсоры используют цветовые фильтры и формируют тройку (R,G,B).

Массив: (height, width, 3), типы uint8/float32. RGB — не единственная модель.

3.2. RGB как инженерная модель

Воспринимаемая яркость — взвешенная сумма каналов (BT.601).

$$Y = 0.299 R + 0.587 G + 0.114 B$$

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Важно: OpenCV использует порядок BGR.

3.3. Переход к HSV

HSV разделяет тон (Hue), насыщенность (S), яркость (V) – удобно для сегментации по цвету.

$$V = \max(R, G, B), \quad S = \frac{V - \min(R, G, B)}{V}$$

$$H = \begin{cases} 60^\circ \frac{G-B}{V-\min}, & V = R \\ 120^\circ + 60^\circ \frac{B-R}{V-\min}, & V = G \\ 240^\circ + 60^\circ \frac{R-G}{V-\min}, & V = B \end{cases}$$

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

3.4. Lab и перцептивная равномерность

Lab: L* — яркость, a* — зелёный ↔ красный, b* — синий ↔ жёлтый; равномерные расстояния ≈ различимости.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M_{\text{RGB} \rightarrow \text{XYZ}} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{cases} L^* = 116 f(Y/Y_n) - 16 \\ a^* = 500 [f(X/X_n) - f(Y/Y_n)] \\ b^* = 200 [f(Y/Y_n) - f(Z/Z_n)] \end{cases}$$

Изменения L* не сдвигают оттенок.

3.5. YCbCr: яркость и цветность

Удобно для кодеков: лума Y и хрома Cb, Cr; субдискретизация 4:2:0.

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = 128 - 0.1687R - 0.3313G + 0.5B$$

$$Cr = 128 + 0.5R - 0.4187G - 0.0813B$$

Глаз менее чувствителен к цветности → эффективная компрессия.

3.6. Конвертации на практике

```
rgb = cv2.imread("flowers.jpg")
hsv = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)
lab = cv2.cvtColor(rgb, cv2.COLOR_BGR2Lab)
ycrcb = cv2.cvtColor(rgb, cv2.COLOR_BGR2YCrCb)

h, s, v = cv2.split(hsv)
cv2.imshow("Hue", h); cv2.imshow("Saturation", s); cv2.imshow("Value", v)
```

В OpenCV Hue кодируется 0–179, что важно для масок.

3.7. Маскирование по цвету

Сегментация: выделяем синие области в HSV с порогами.

```
lower_blue = np.array([100,150,0])
upper_blue = np.array([140,255,255])
mask = cv2.inRange(hsv, lower_blue, upper_blue)
result = cv2.bitwise_and(rgb, rgb, mask=mask)
```

Базис робототехники, видеоаналитики и контроля качества.

3.8. Точность и битность

Наука/HDR/медицина — 12–16 бит на канал; для отображения требуется масштабирование.

```
img16 = cv2.imread("microscopy.tif", cv2.IMREAD_UNCHANGED)
img8 = cv2.convertScaleAbs(img16, alpha=255.0/65535.0)
```

Без масштабирования 16-битные кадры выглядят чёрными.

3.9. Геометрия цветовых пространств

RGB — куб, HSV — цилиндр (угол=тон, радиус=насыщенность, высота=яркость), Lab — близок к сфере.

Коррекция и стилизация интерпретируются как геометрические преобразования.

3.10. Выбор пространства — успех обработки

RGB — визуализация, HSV — пороги по цвету, Lab — текстуры/контраст, YCbCr — кодирование.

Преобразования сохраняют геометрию, но меняют чувствительность к освещению.

Раздел 4. Форматы и структура изображения

4.1. Файл: данные и метаданные

Изображение как контейнер: заголовок (размеры, каналы, битность, компрессия) + тело пикселей, метаданные (EXIF).

Порядок записи и стандарты формата определяют интерпретацию.

4.2. BMP – без компрессии

Простая структура, явное хранение пикселей. Размер:

$$S = \frac{W \cdot H \cdot n}{8} \text{ байт}$$

Просто для отладки, но неэкономично (например, 1920×1080×24 бита \approx 6 МБ).

4.3. PNG — сжатие без потерь

Этапы: фильтрация (предсказание пикселя) \rightarrow DEFLATE (LZ77+Хаффман).

$$R(x, y) = P(x, y) - \hat{P}(x, y)$$

Остатки снижают энтропию; точная реконструкция — для медицинских/научных данных и UI.

4.4. JPEG – с потерями (DCT)

YCbCr → блоки 8×8 → DCT → квантование → энтропийное кодирование.

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

Квантование матрицей Q(u,v) подавляет высокие частоты (детали).

4.5. Энтропийное кодирование (Хаффман)

После квантования много нулей; длина кода $\sim -\log_2 p_i$, средняя длина — энтропия.

$$H = - \sum_i p_i \log_2 p_i$$

JPEG даёт 10–20× сжатие; при сильной компрессии — блоковые артефакты.

4.6. Декодирование JPEG

Чтение заголовков → декод Хаффмана → обратное DCT → обратное YCbCr→RGB.

```
cv2.imwrite("compressed.jpg", img, [cv2.IMWRITE_JPEG_QUALITY, 40])
restored = cv2.imread("compressed.jpg")
diff = cv2.absdiff(img, restored)
cv2.imshow("Difference", diff)
```

Видны блоки и потеря высоких частот.

4.7. Альфа-канал и прозрачность

RGBA: α — непрозрачность (0 — прозрачный, 255 — непрозрачный). Композитинг:

$$I_{\text{out}} = \alpha I_{\text{fg}} + (1 - \alpha) I_{\text{bg}}$$

Широко в графике, интерфейсах, AR.

4.8. Метаданные EXIF

Хранят параметры съёмки (диафрагма, ISO, выдержка, модель, дата) — полезно для коррекции.

```
from PIL import Image
img = Image.open("photo.jpg")
exif = img.getexif()
print(exif.get(306)) # Дата съемки
```

Делает файл самодостаточным источником контекста.

4.9. Потери и устойчивость форматов

BMP устойчив, но велик; PNG без потерь и устойчив; JPEG — компромисс и деградирует при пересохранении.

Выбор под задачу: архив/наука — PNG/TIFF; публикации — JPEG; промежуточное — BMP/RAW.

4.10. Компрессия как фильтрация

JPEG — фактически низкочастотное сглаживание; иногда полезно для крупных структур.

Чрезмерные потери ухудшают точность CV; учитывайте спектральные изменения источника.

Раздел 5. Пространственные свойства и фильтрация

5.1. Локальность и окрестность

Новое значение пикселя определяется окрестностью $(2k+1) \times (2k+1)$ с весами $w(u,v)$.

$$I'(x, y) = \sum_{u=-k}^k \sum_{v=-k}^k w(u, v) I(x - u, y - v)$$

Локальные фильтры – основа обработки.

5.2. Свёртка и корреляция

Свёртка переворачивает ядро, корреляция — нет; при симметрии совпадают.

$$(I * K)(x, y) = \sum_{u=-m}^m \sum_{v=-n}^n I(x - u, y - v) K(u, v)$$

Тип ядра задаёт: сглаживание, усиление, дифференцирование.

5.3. Сглаживающие фильтры

Усредняющий фильтр снижает шум, но размывает границы.

$$K(u, v) = \frac{1}{(2k + 1)^2}$$

Гауссов фильтр даёт больше веса центру:

$$K(u, v) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right)$$

```
blur = cv2.GaussianBlur(img, (5,5), 1.0)
```

5.4. Высокие частоты и контуры

Лапласиан — вторая производная, подчёркивает границы.

$$K_{\text{Lap}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```
kernel = np.array([[1,1,1],[1,-8,1],[1,1,1]], np.float32)
edges = cv2.filter2D(img, -1, kernel)
```

База детекторов (Sobel, Prewitt, Canny) и CNN.

5.5. Повышение резкости (Unsharp)

Размываем I_b и добавляем высокочастотную маску.

$$I'(x, y) = I(x, y) + \lambda (I(x, y) - I_b(x, y))$$

```
blur = cv2.GaussianBlur(img, (5,5), 2)
sharp = cv2.addWeighted(img, 1.5, blur, -0.5, 0)
```

Усиливает края без значительного шума.

5.6. Нелинейные фильтры

Медианный устраниет «соль и перец» без размывания границ.

$$I'(x, y) = \text{median}\{I(x + u, y + v) : (u, v) \in \Omega\}$$

```
median = cv2.medianBlur(img, 5)
```

Билатеральный сохраняет края, учитывая расстояние и разницу яркости.

$$I' = \frac{1}{W} \sum_{\Omega} I(u, v) \exp\left(-\frac{d^2}{2\sigma_s^2} - \frac{\Delta I^2}{2\sigma_r^2}\right)$$

5.7. Градиент и производные

Градиент даёт величину и направление изменений.

$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right]$$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

```
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1)
mag = cv2.magnitude(gx, gy)
```

5.8. Частотная интерпретация фильтров

Фильтр \Leftrightarrow частотная характеристика $H(u,v)$, умножаемая на спектр $F(u,v)$.

$$F(u, v) = \sum_{x,y} I(x, y) e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

```
f = np.fft.fft2(img); fshift = np.fft.fftshift(f)
magnitude = 20*np.log(np.abs(fshift)+1)
```

Центр — низкие частоты, края — детали.

5.9. Масштабирование и интерполяция

Изменение размера требует интерполяции; при уменьшении нужен антиалиасинг.

```
resized = cv2.resize(img, (640,480), interpolation=cv2.INTER_LINEAR)
```

Альтернативы: bicubic, Lanczos — лучшее качество при увеличении.

5.10. Локальные фильтры и CNN

Свёртки в CNN — та же операция, но веса обучаются из данных.

Классическая теория фильтрации — фундамент понимания архитектуры CV.

Раздел 6. Частотное представление

6.1. Переход к частотному взгляду

2D-ДПФ: модуль — амплитуда, аргумент — фаза; обратное ДПФ восстанавливает изображение.

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y) e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

$$I(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

6.2. Энергия и спектр

Плавные области → энергия в низких частотах; текстуры/шум — в высоких.

$$S(u, v) = \log(1 + |F(u, v)|)$$

```
f = np.fft.fft2(img); fshift = np.fft.fftshift(f)
magnitude = 20*np.log(np.abs(fshift)+1)
plt.imshow(magnitude, cmap='gray')
```

fftshift центрирует нулевую частоту.

6.3. Свёртка умножение в спектре

Фильтр в пространстве соответствует умножению спектров — выгодно для больших ядер.

$$\mathcal{F}\{I * K\} = F_I \cdot F_K$$

Гаусс подавляет высокие частоты; лапласиан — усиливает.

6.4. Идеальные НЧ/ВЧ фильтры

Идеальный НЧ: пропускает частоты с $D(u,v) \leq D_0$; ВЧ — наоборот.

$$H(u, v) = \begin{cases} 1, & D(u, v) \leq D_0 \\ 0, & D(u, v) > D_0 \end{cases}, \quad D(u, v) = \sqrt{(u - \frac{M}{2})^2 + (v - \frac{N}{2})^2}$$

```
rows, cols = img.shape
mask = np.zeros((rows, cols), np.uint8)
r = 50; cv2.circle(mask, (cols//2, rows//2), r, 1, -1)
fshift = np.fft.fftshift(np.fft.fft2(img))
res = np.abs(np.fft.ifft2(np.fft.ifftshift(fshift*mask)))
```

6.5. Гауссовые фильтры в спектре

Гладкая маска без резких границ избегает эффекта Гиббса.

$$H(u, v) = \exp\left(-\frac{D(u, v)^2}{2D_0^2}\right), \quad H_{\text{high}} = 1 - H_{\text{low}}$$

Удобно для контролируемого сглаживания перед детекцией контуров.

6.6. Фаза против амплитуды

Структура кодируется фазой; амплитуда — контраст. Полная реконструкция требует обоих.

$$F(u, v) = A(u, v) e^{j\phi(u, v)}, \quad A = |F|, \phi = \arg F$$

Замена фаз на случайные разрушает форму, сохраняя общий тон.

6.7. Фильтрация в частотной области

Шаги: FFT → умножение на $H(u,v)$ → обратная FFT. Эффективно на больших изображениях.

```
f = np.fft.fft2(img); fshift = np.fft.fftshift(f)
mask = np.zeros(img.shape, np.uint8)
r = 40; cv2.circle(mask, (img.shape[1]//2, img.shape[0]//2), r, 1, -1)
filtered = fshift * mask
img_back = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered)))
```

Гомоморфная фильтрация: подавление НЧ освещённости в лог-домене.

6.9. Частоты и зрение

Зрение чувствительно к средним пространственным частотам ($\approx 2\text{--}6$ циклов/градус); к очень низким/высоким — меньше.

Оттого умеренная потеря высоких частот (JPEG) малозаметна. Частотный анализ объясняет восприятие деталей, контраста и резкости.