پیاده‌سازی روش CALIC برای فشرده‌سازی بی‌تلف تصویر

عباس حاتمی خوشمردان

https://github.com/khoshmard/calic/

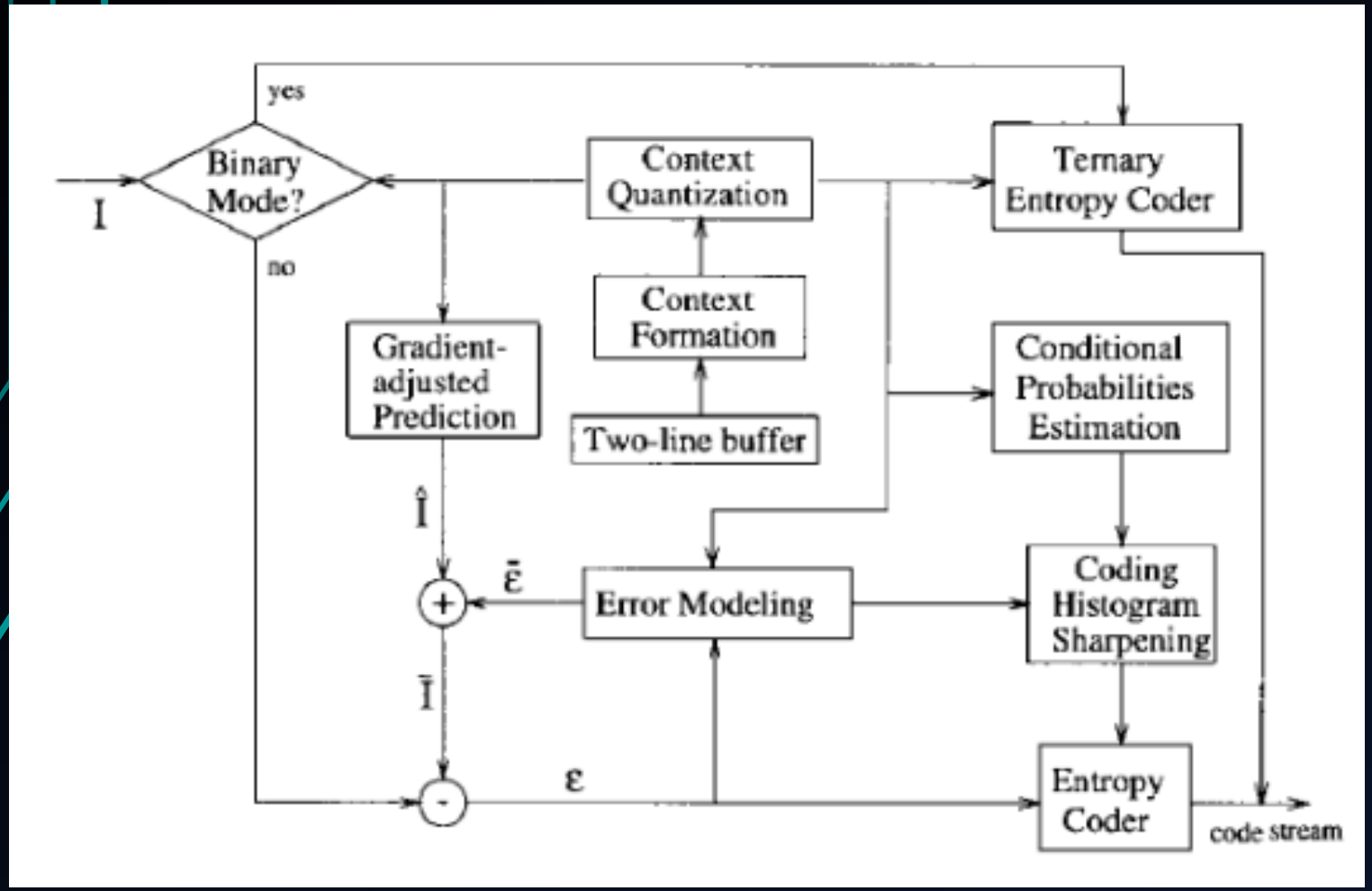
تاریخچه

- کمیته‌ی JPEG در سال ۱۹۹۴ برای پیدا کردن الگوریتم بهتری برای فشرده‌سازی بی‌تلف تصویر ثابت درخواست ارائه پیشنهاده کرد[۱].
- ۹ الگوریتم با توجه به شرایط اعلام شده‌ی کمیته ارائه شد.
- در بررسی الگوریتم CALIC بهترین عملکرد را دارا بود [۲].
- آقایان «ژیائولینگ وو» و «نصیر ممون» الگوریتم CALIC را پیشنهاد دادند [۳].
- CALIC مخفف «کدک تصویر بی‌تلف وفقی زمینه‌محور» است.

الگوریتم CALIC

- فقط به یک بار پایش تصویر نیاز دارد
- از پیش‌بینی و الگوی زمینه همزمان استفاده می‌کند
- پیش‌بینی و الگوی زمینه فقط به دو خط پایش شده‌ی قبلی تصویر بستگی دارند
- در پیاده‌سازی نیاز به یک بافر ساده دارد
- دو حالت دودویی و پیوسته دارد [۳]

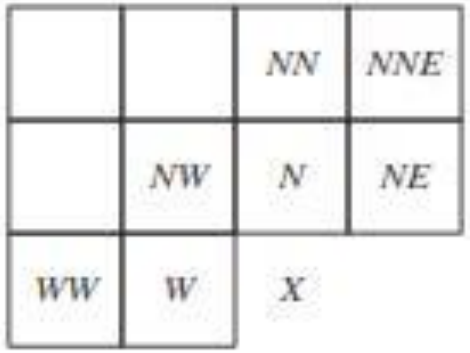
اجزای اصلی الگوریتم CALIC



- پیش‌بینی تنظیم‌شده با گرادیان
- انتخاب زمینه و چندی‌سازی
- مدل‌سازی خطای پیش‌بینی با زمینه
- کدگذاری آنتروپی خطای پیش‌بینی [۴]

پیش‌بینی تنظیم‌شده با گردیان

- می‌خواهیم پیش‌بینی $\hat{X}$ را از روی X پیدا کنیم.
- دو گردایان عمودی و افقی را تشکیل می‌دهیم.
- با توجه به مقدار این گرادیان‌ها پیش‌بینی را انتخاب می‌کنیم [۵]



$$d_h = |W - WW| + |N - NW| + |NE - N|$$
$$d_v = |W - NW| + |N - NN| + |NE - NNE|$$

```
if d_h − d_v > 80
    X̂ ← N
else if d_v − d_h > 80
    X̂ ← W
else
{
    X̂ ← (N + W)/2 + (NE − NW)/4
    if d_h − d_v > 32
        X̂ ← (X̂ + N)/2
    else if d_v − d_h > 32
        X̂ ← (X̂ + W)/2
    else if d_h − d_v > 8
        X̂ ← (3X̂ + N)/4
    else if d_v − d_h > 8
        X̂ ← (3X̂ + W)/4
}
```

پیش‌بینی تنظیم‌شده با گردیان

- پیاده‌سازی بخش GAP در پایتون
- فایل gap.py

```
10    def GAP(im, i, j):
12        Iw = get(im, i-1, j)
13        Ine = get(im, i+1, j-1)
14        Inw = get(im, i-1, j-1)
15        Inn = get(im, i, j-2)
16        Iww = get(im, i-2, j)
17        Inne = get(im, i+1, j-2)
18        dh = abs(Iw - Iww) + abs(In - Inw) + abs(In - Ine)
19        dv = abs(Iw - Inw) + abs(In - Inn) + abs(Ine - Inne)
20        if dv - dh > 80: #sharp horizontal edge
21            return Iw
22        elif dv - dh < -80: #sharp vertical edge
23            return In
24        else:
25            ic = (Iw + In) / 2 + (Ine - Inw) / 4
26            if dv - dh > 32: # horizontal edge
27                return (ic + Iw) / 2
28            elif dv - dh > 8: #weak horizontal edge
29                return (3*ic + Iw) / 4
30            elif dv - dh < -32: # vertical edge
31                return (ic + In) / 2
32            elif dv - dh < -8: # weak vertical edge
33                return (3*ic + In) / 4
34            return ic
```

```
def get(im, i, j):
    if 0 <= i < im.shape[0] and 0 <= j < im.shape[1]:
        return im[i, j]
    return 0
```

```
37    im = data.camera() # is gray scale image.
38    out = np.empty((im.shape[0], im.shape[1])) # Icap
39
40    for i in range(im.shape[0]):
41        for j in range(im.shape[1]):
42            out[i, j] = GAP(im, i, j)
```

پیش‌بینی تنظیم‌شده با گردیان

- مقایسه خروجی تابع GAP با تصویر ورودی

انتخاب زمینه و چندی‌سازی

$$\delta = d_h + d_v + 2\left|N - \widehat{N}\right|$$

$$\begin{aligned}
0 \leqslant \delta < q_1 &\Rightarrow \text{Context } 1 \\
q_1 \leqslant \delta < q_2 &\Rightarrow \text{Context } 2 \\
q_2 \leqslant \delta < q_3 &\Rightarrow \text{Context } 3 \\
q_3 \leqslant \delta < q_4 &\Rightarrow \text{Context } 4 \\
q_4 \leqslant \delta < q_5 &\Rightarrow \text{Context } 5 \\
q_5 \leqslant \delta < q_6 &\Rightarrow \text{Context } 6 \\
q_6 \leqslant \delta < q_7 &\Rightarrow \text{Context } 7 \\
q_7 \leqslant \delta < q_8 &\Rightarrow \text{Context } 8
\end{aligned}$$

- فقط محاسبه GAP افزونگی آماری را کاملاً از بین نمی‌برد.
- پراکنش خطای پیش‌بینی با نرمی تصویر در اطراف پیکسل پیش‌بینی همبستگی زیادی دارد.
- برای مدل‌سازی این همبستگی δرا محاسبه می‌کنیم.
- مقدار δرا چندی‌سازی می‌کنیم و بر اساس این چندی‌سازی زمینه را تعیین می‌کنیم
- در عمل تعداد سطوح چندی‌سازی را ۸ می‌گیریم
- همچنین مقدار پیش‌بینی را با مقادیر زیر مقایسه می‌کنیم [۵]

$$[N, W, NW, NE, NN, WW, 2N - NN, 2W - WW]$$

انتخاب زمینه و چندی‌سازی

```
# Texture Quantizer
temp = list(map(lambda x: int(x < ic),[(2*Iw)-Iww,(2*In)-Inn,Iww,Inn,Ine,Inw,Iw,In]))
B = temp[0] << 7 | temp[1] << 6 | temp[2] << 5 | temp[3] << 4 | temp[4] << 3 | temp[5] << 2 | temp[6] << 1 | temp[7]
```

```
# Delta.
global err_for_del
delt = dh + dv + 2*abs(err_for_del) #Error energy estimator computation

# Error Energy Quantizer
#   Now quantize error energy estimator according to thresholds given by CALIC
#   Into 8 partitions
Qdel = -1
k = 0
while k < len(thre):
    if delt <= thre[k]:
        Qdel = k
        break
    k += 1
if Qdel == -1:
    Qdel = 7
```

$$b_k = \begin{cases} 0 & \text{if } x_k \geq \dot{I}[i,j] \\ 1 & \text{if } x_k < \dot{I}[i,j] \end{cases}, \quad 0 \leq k < K = 8.$$

```
# Context Modeling Context C
C = B * Qdel // 2
```

مدل‌سازی خطای پیش‌بینی با زمینه

```
# global err
# Update N (No of occurrences)
N[C] += 1
S[C] += err_for_del
# Limit the count
if N[C] == 255:
    N[C] = N[C] / 2
    S[C] = S[C] / 2

ed = S[C] // N[C]
Itilde = ic + ed
out[i, j] = Itilde
context[i, j] = C  # store the context
err_for_del = get(im, i, j) - Itilde
```

- می‌توان بخشی از خطای پیش‌بینی را با زمینه کاهش داد

کدگذاری آنتروپی خطای پیش‌بینی

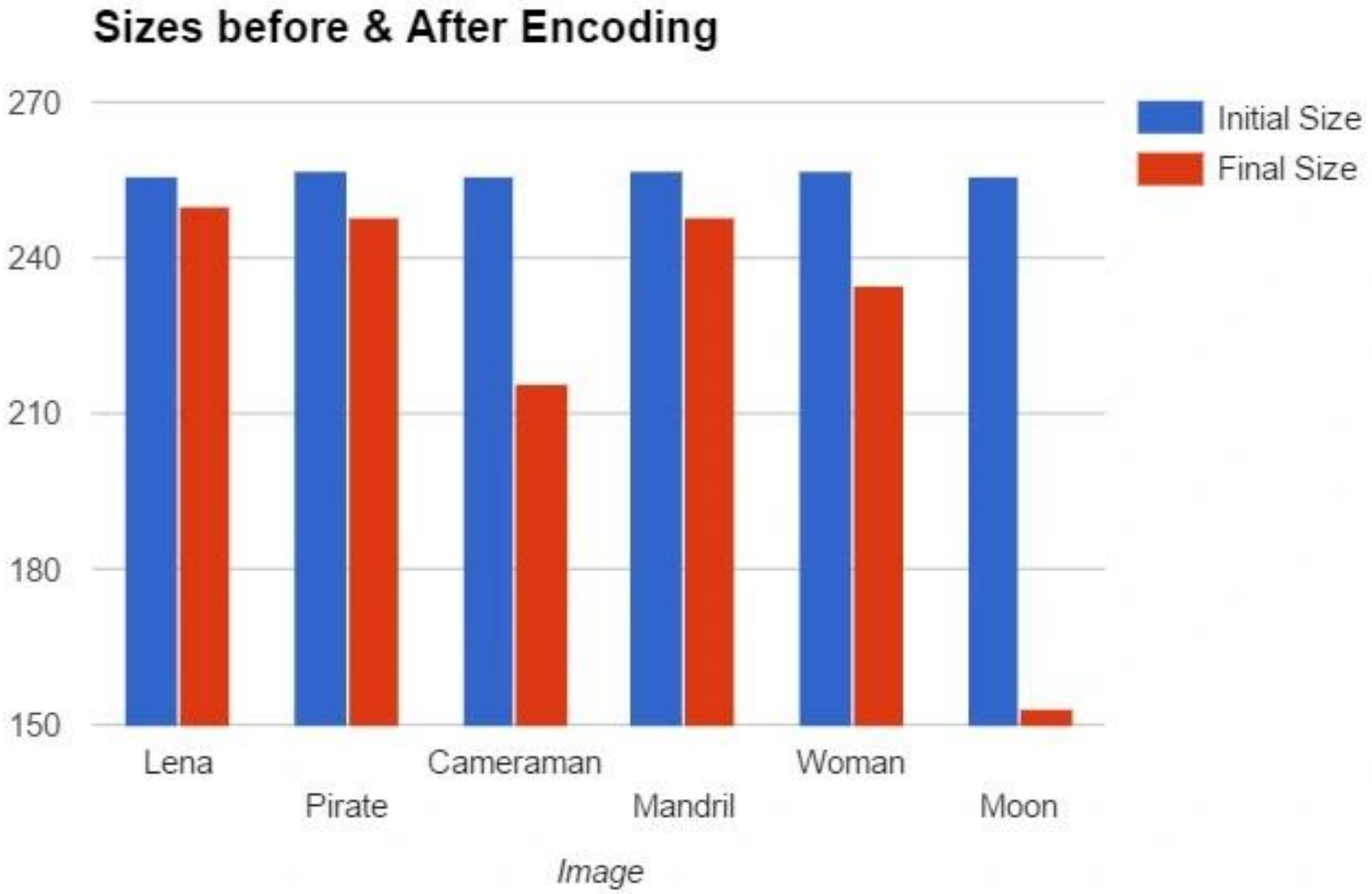
- در [۳] نوع کدگذاری آنتروپی تعیین نشده است.
- کدگذاری آنتروپی می‌تواند حسابی یا هافمن تطبیقی باشد.
- در این پیاده‌سازی از کدگذاری حسابی استفاده شده است.

- https://www.nayuki.io/page/reference-arithmetic-coding
- فایل‌های arithmetic.py ، arithmeticcoding.py و arithmetic-adaptive-coding.py

مراحل انجام کار

- در فایل calic.ipynb الگوریتم فراخوانی شده است.
- فایل‌های عکس فرخوانی می‌شوند و عکس خام آنها با فرمت raw ذخیره می‌شوند
- مراحل کدگذاری روی آنها انجام می‌شود
- با فرمت craw ذخیره می‌شوند
- کدگذاری حسابی روی فایل craw انجام می‌شود
- فایل نهایی با فرمت calic ذخیره می‌شوند

نتایج


Sizes before & After Encoding
270
240
210
180
150
Initial Size
Final Size
Lena
Pirate
Cameraman
Mandril
Woman
Moon
Image

کارهای آینده

- الگوریتم CALIC در بعضی از محتواها بهتر از دیگر محتواها عمل می‌کند
- می‌توان بخشی به الگوریتم افزود که بر اساس محتوا الگوریتم را متناسب کند.
- می‌توان پارامترهای بهینه شده را از با استفاده از یادگیری ماشین پیدا کرد.
- می‌توان از کدگذاری Golomb-Rice نیز برای فشرده‌سازی خطای پیش‌بینی استفاده کرد.
- مکانیزیم فیدبک خطا پیاده‌سازی نشده است.

از توجه شما سپاسگذارم

منابع

- [1] ISO/IEC JTC 1/SC 29/WG 1. Call for contributions - lossless compression of continuous-tone still pictures. ISO Working Document ISO/IEC JTC1/SC29/WG1 N41, March 1994.
- [2] S. Urban. Compression results - lossless, lossy ±1, lossy ±3. ISO Working Document ISO/IEC JTC1/SC29/WG1 N281, 1995.
- [3] Xiaolin Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, Atlanta, GA, USA, 1996, pp. 1890-1893 vol. 4, doi: 10.1109/ICASSP.1996.544819.
- [4] Rashid Ansari and Nasir Memon, "The JPEG Lossless Standards", January 18, 1999
- [5] K. Sayood. Lossless Image Compression. In: Introduction to Data Compression. Morgan Kaufmann Publishers, San Francisco, CA, USA, 4th edition.