دانشگاه تهران

دانشکده علوم و فنون نوین

تمرین دوم

نام و نام خانوادگی	فاطمه چیت ساز
شماره دانشجویی	830402092
تاریخ ارسال گزارش	2 دی 1402

Contents

تاثیر تفاضل گیری در هافمن

در کدینگ هافمن، به هر نماد یک کد بیت اختصاص داده می شود. طول کدها به تناسب فرکانس نمادها تعیین می شود: نمادهای با فرکانس بالا کدهای کوتاهتر و نمادهای با فرکانس پایین کدهای بلندتر دریافت می کنند. این امر به طور کلی فشرده سازی خوبی را ارائه می دهد، زیرا نمادهای رایج با بیت های کمتری نشان داده می شوند.

با این حال، می توان با استفاده از تکنیکی به نام "تفاضل گیری" فشرده سازی را بیشتر بهبود بخشید. تفاضل گیری قبل از اعمال کدینگ هافمن، مقادیر داده را به جای خودشان، به صورت اختلاف بین مقادیر متوالی ذخیره می کند.

مزیت تفاضل گیری:

کاهش دامنه مقادیر: مقادیر تفاضلی معمولاً دامنه کوچکتری نسبت به مقادیر اصلی دارند. این به نوبه خود باعث می شود که کدهای هافمن کوتاهتر شوند، زیرا تعداد بیت های مورد نیاز برای نشان دادن مقادیر کوچکتر کمتر است.

افزایش فرکانس نمادهای مشابه: در داده های تفاضلی، نمادهای مشابه (مانند 0 یا 1) با فرکانس بیشتری نسبت به داده های اصلی ظاهر می شوند. کدینگ هافمن از این امر بهره می برد و کدهای کوتاهتری به این نمادها اختصاص می دهد.

البته یک نکته قابل توجه است که تفاضل گیری همیشه فشرده سازی را افزایش نمی دهد. در برخی موارد، مانند زمانی که داده ها تصادفی هستند، تفاضل گیری ممکن است فشرده سازی را بدتر کند.

بررسی تفاضل در تصویر

تصویری از خودتان تهیه کنید و مؤلفه روشنایی (Y) آن را استخراج کنید. بررسی کنید که داده های تفاضل پیکسل های تصویر شما چه وضعی دارند؟

در اینجا ما ابتدا تصویری داریم و قصد داریم آرگومان روشنایی را از تصویر استخراج کرده و سپس هیستوگرام آن را نمایش دهیم سپس فرایند تفاضی گیری را نیز انجام دهیم و باز هیستوگرام را در تصویر خود مشاهده کنیم برای این کار ابتدا

تصویر رو میخونیم:

```
imagePath = 'my_image.jpg';

% Read the image
rgbImage = imread(imagePath);
```

تصویر رو به فضای رنگ YCbCr تبدیل می‌کنیم:

```
% Convert the image to YCbCr color space
ycbcrImage = rgb2ycbcr(rgbImage);
```

مؤلفه روشنایی (Y) رو استخراج می‌کنیم:

```
% Extract the luminance component Y
YComponent = ycbcrImage(:,:,1);
```

نمایش تصویر اصلی و هیستوگرام آن:

```
% Display the original image and its histogram
figure;
subplot(2,2,1);
imshow(rgbImage);
title('Original Image');
```

```
% Display the Y component and its histogram before computing differences
subplot(2,2,3);
imshow(YComponent, []);
title('Y Component Before Differences');
```

```
subplot(2,2,4);
imhist(YComponent);
title('Histogram of Y Component Before Differences');
```

نمایش مؤلفه روشنایی (Y) قبل از محاسبه تفاضل‌ها:

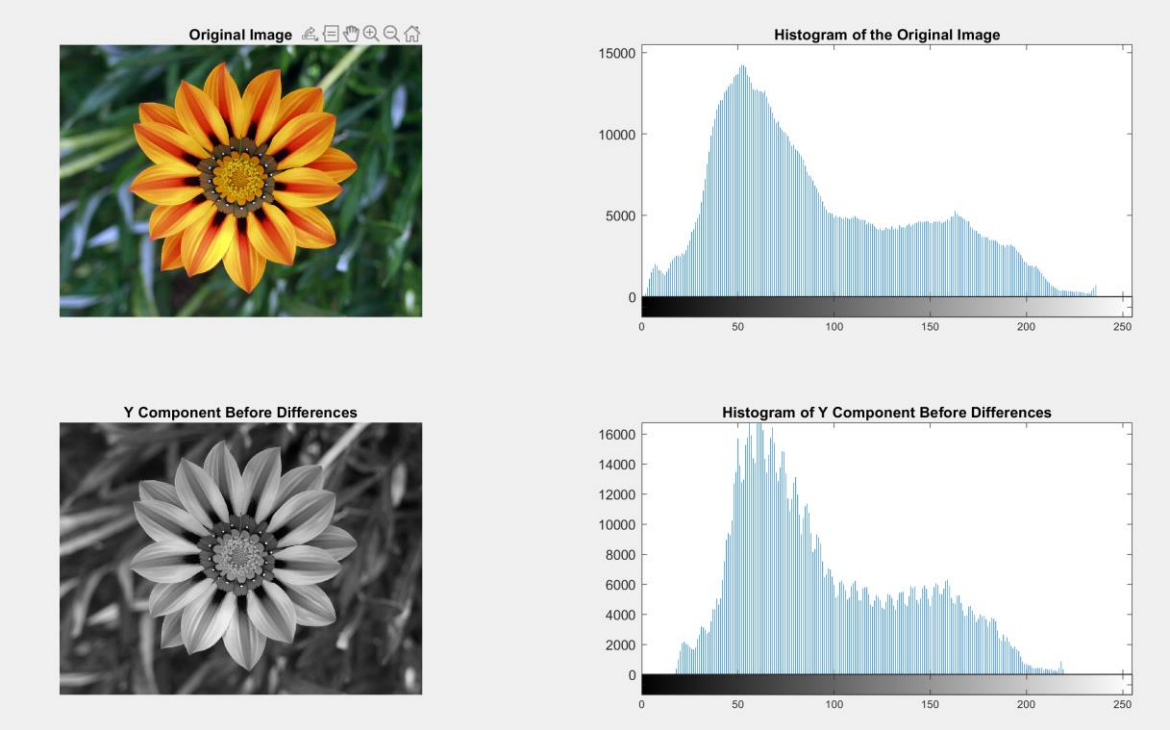
```
% Compute pixel differences
pixelDifferences = diff(double(YComponent), 1, 2);
```

محاسبه تفاضل‌های پیکسل و نمایش نتایج:

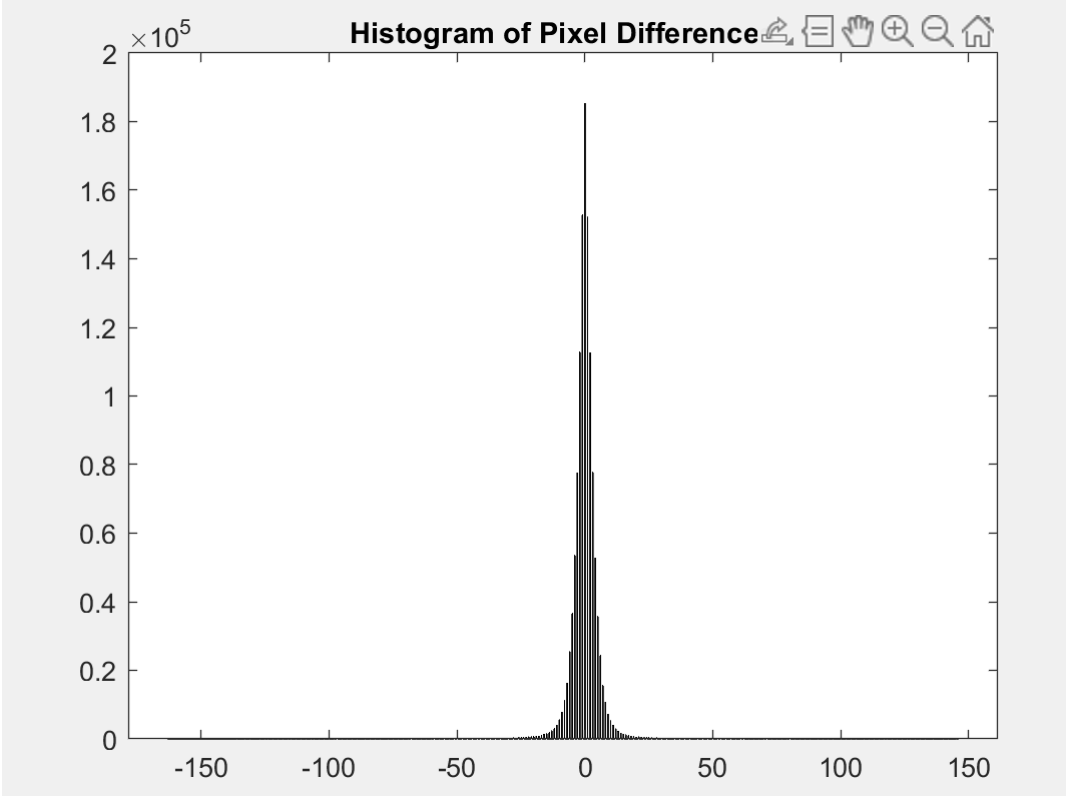
```
% Display histogram after computing differences
figure;
subplot(1,1,1);
histogram(pixelDifferences);
title('Histogram of Pixel Differences');
```

خروجی کد :

تصویر اصلی و هیستوگرام آن :



اعمال diff بر روی تصویر



مشاهده میشود محدوده تغیرات بسیار کوچکتر شده است و دلیل این امر نزدیک بودن مقادیر پیکسل های همسایه هم هست و در واقع diff ها مقادیر کوچکتر با تکرار بیشتری دارند

تصویری پیدا کنید که برعکس باشد و تفاضل گیری باعث شود که فشرده سازی هافمن کمتر (بدتر) بشود. به کمک MATLAB نشان دهید که تصویر شما چنین حالتی دارد؟

در زمانی که تصاویر ما دارای جزئیات فراوان باشند و تصاویر سخت و پیچیده ای باشند فرایند تفاضل گیری باعث بدتر شدن فرایند فشرده سازی هافمن است

مثلا برای اینکار ما یک تصویر رندوم از اعداد رندوم ایجاد کردیم

```
imageSize = 256;
randomDifferences = randi([-10, 10], 1, 10);
I_gray = cumsum(randomDifferences, 2);
```

سپس فرایند diff گرفتن را روی آن انجام دادیم

```
I_diff =  [I_gray(:, 1), diff(I_gray, 1, 2)]; % Compute the difference between adjacent pixels
```

تصویر اصلی :

x10 double

1	2	3	4	5	6	7	8	9	10
5	14	7	-1	-2	5	3	5	12	17

تصویر بعد از فرایند تفاضل گیری:

x10 double

1	2	3	4	5	6	7	8	9	10
5	9	-7	-8	-1	7	-2	2	7	5

بعد فرایند فشرده سازی را هم روی تصویر اصلی هم روی تصویر حاصل از تفاضل انجام میدهیم

```
% Huffman encoding
I_gray = double(I_gray);
symbols = unique(I_gray(:));
counts = hist(I_gray(:), symbols);
prob = counts / sum(counts);
dict = huffmandict(symbols, prob);
comp = huffmanenco(I_gray(:), dict);
```

در نهایت نیز اندازه تصویر اصلی تصویر اصلی بعد فشرده سازی و تصویر تفاضلی بعد فشرده سازی را نمایش میدهیم

```
% Display the original and compressed image sizes
disp(['Original image size: ', num2str(numel(I_gray)*8), ' bits']);
disp(['Compressed image size: ', num2str(numel(comp)), ' bits']);
disp(['Compressed diff image size: ', num2str(numel(comp_diff)), ' bits']);
```

خروجی:

```
>> untitled
Original image size: 80 bits
Compressed image size: 29 bits
Compressed diff image size: 30 bits
```

همینطور که مشاهده میشود فشرده سازی عادی بهتر بوده است

برسی فشرده سازی تصویر با استفاده از کدینگ هافمن

چرا در جدول 3.29، تصویر کره زمین خیلی بهتر از سه تصویر دیگر فشرده شده است؟

این موضوع به احتمال زیاد به دلیل توزیع مقادیر پیکسل ها در تصویر کره زمین است. اگر توزیع مقادیر پیکسل ها نامتوازن باشد، یعنی برخی مقادیر بیشتر از بقیه تکرار شوند، Huffman coding می‌تواند به خوبی از این الگوی تکرار بهره ببرد و به فشرده سازی بهتر منجر شود. تصویر کره زمین احتمالاً دارای نواحی وسیعی با مقادیر پیکسل یکسان است که باعث می‌شود فشرده سازی بهتری نسبت به تصاویری با تغییرات پیکسلی بیشتر داشته باشد البته نکته دیگری نیز قابل توجه است که بک گراند این تصویر رنگ مشکی دارد بنابراین با توجه به توزیع فراوان این رنگ کدینگ هافمن میتواند کد های کوچکتری به آن دهد و بنابراین تصویر بهتر فشرده شود

در مقایسه جدول 3.29 با 3.30، چرا فشرده سازی سه تا از تصاویر خیلی بهتر شده اما برای تصویر نقشه هوایی بهبود کمی حاصل شده است؟

در جدول 3.30 از تکنیک تفاضل گیری بین پیکسل‌های همسایه استفاده شده است. این تکنیک زمانی موثرتر است که پیکسل‌های همسایه شباهت زیادی به هم داشته باشند، مانند تصاویر پسربچه‌ها و کره زمین که نواحی هموارتری دارند. اما در تصویر نقشه هوایی، تغییرات بین پیکسل‌های همسایه بیشتر

است و تفاضل گیری نمی‌تواند آن‌ها را به مقادیر تکراری زیادی تبدیل کند. در نتیجه، بهبود حاصل از این تکنیک برای تصویر نقشه هوایی کمتر است.

چرا در جدول 3.31، هافمن وفقی نسبت به هافمن معمولی، در تصویر کره زمین «بدتر» عمل کرده است؟ (جدول 3.31 را با جدول 3.30 مقایسه کنید)

این موضوع می‌تواند به این دلیل باشد که هافمن وفقی به صورت تک‌گذر عمل می‌کند و در حین فشرده سازی، جدول کدها را به تدریج تنظیم می‌کند. این موضوع ممکن است در برخی موارد باعث شود که کدهای بهینه تولید نشوند، مخصوصاً اگر توزیع مقادیر پیکسل ها در تصویر به طور ناگهانی تغییر کند.(مثل همین که رنگ بک گراند مشکی به رنگ کره زمین تغیر میکنه

در مقابل، هافمن معمولی در دو گذر عمل می‌کند و در گذر اول، یک جدول کدهای بهینه بر اساس توزیع کامل مقادیر پیکسل ها ایجاد می‌کند. این موضوع باعث می‌شود که هافمن معمولی در برخی موارد، مانند تصویر کره زمین، عملکرد بهتری نسبت به هافمن وفقی داشته باشد.