آرین تشکر - 40023494

تکلیف سوم درس سیستم های چند رسانه ای پیشرفته

ساختار فایل تحویلی:

پوشه ی اصلی فایل تکلیف شامل یک فایل Report.pdf (فایلی که هم اکنون در حال خواندن آن هستید) و یک پوشه به نام Deliverables می باشد. در پوشه ی Deliverables چندین فایل سورس متلب قرار گرفته است که هر کدام از آن ها برای قسمتی از تکلیف مورد استفاده قرار می گیرند (به خصوص فایل main.m شامل کد تست بخش های مختلف می باشد). همچنین در این پوشه، دو پوشه ی دیگر به نام های Inputs و Results قرار دارند که به ترتیب فایل های ورودی و نتایج را در خود جای می دهند.

تمامی فایل به صورت کامل کامنت گذاری شده اند، در صورت نیاز می توانید به اصل کد ها برای نکات پیاده سازی و توضیحات تکمیلی مراجعه کنید.

سوال 1) رسم تصاویر پایه ی DCT:

سورس کد های مربوط: Deliverables\DCT_Basis.m و Deliverables\Alpha.m

کد ها:

DCT_Basis.m:

```
function dct_basis = DCT_Basis(block_size, resolution)
    % dct_basis[i, j] -> a size by size representation of the (i, j) basis
    % function.
    dct_basis = zeros([block_size, block_size, resolution, resolution]);
    for r=1:block_size
        for c=1:block_size
            % basis function at (r, c)
            for i=1:resolution
                for j=1:resolution
                    dct_basis(r, c, i, j) = Alpha(r-1, resolution)*...
                        Alpha(c-1, resolution)*...
                        cos((pi*(2*(i-1)+1)*(r-1))/(2*resolution))*...
                        cos((pi*(2*(j-1)+1)*(c-1))/(2*resolution));
                end
            end
        end
    end
end
```

Alpha.m:

```
function alpha = Alpha(i, size)
    if i == 0
        alpha = sqrt(1/size);
    else
        alpha = sqrt(2/size);
    end
end
```

پیاده سازی این قسمت نکته ی خاصی ندارد و مستقیماً از روی فرمول های مربوط به محاسبه ی DCT نوشته شده است.

سوال 2) بررسی صفحات بیتی تصویر DCT:

سورس کد مربوط: Deliverables\main.m

نتایج هر یک از موارد خواسته شده با اجرای main.m بدست می آیند. در مورد علت پدیده ی مشاهده شده در جلسه 25 می توان گفت که از آنجایی که ضرایب DCT در بالا سمت چپ تصویر دارای مقادیر بزرگتری هستند (ضرایب DC و AC های کم فرکانس) از بیت های 1 در صفحات بیتی بالاتری برای نمایش دادن آن ها باید استفاده شود و دلیل این که 1 های بیشتری در بالا سمت چپ تصویر مشاهده می شوند می تواند این موضوع باشد.

در مورد تعویض صفحات بیتی DCT و تصویر اصلی با یک تصویر باینری تصادفی می توان گفت که به نظر می رسد که در صفحات بیتی پایین، در هر دو مورد نواحی با فرکانس بالا تحت تاثیر بیشتری قرار می گیرند و هر چه بالاتر می رویم نواحی فرکانس پایین در هر دو تصویر بیشتر تغییر می کنند. علت این که چرا نواحی کم فرکانس با جایگزین کردن تصویر باینری تصادفی در صفحات بیتی بالای DCT تحت تاثیر قرار می گیرند احتمالاً به پاسخ قسمت اول همین سوال بر می گردد.

سوال 3) Steganography:

سورس کد های مربوط: Deliverables\StegHide.m

این فایل شامل یک کلاس ساده برای انجام embedding و extraction یک فایل دلخواه درون تصویر می باشد. برای انجام embedding ماهیت تصویر بودن فایل ورودی در نظر گرفته نشده است و فایل به صورت بایت به بایت خوانده می شود.

کدها:

StegHide::StegHide:

```
function obj = StegHide(cover, secret)
            % Embeds secret into cover
            if nargin > 0
                obj.cover_image = imread(cover);
                handle = fopen(secret);
                % bit-level read
                secret_bits = uint8(fread(handle, '*ubit1'));
                fclose(handle);
                % secret key is equal to the length of secret message
                obj.internal_key = numel(secret_bits);
                % construct lsb as a vector
                lsb = uint8(zeros([numel(obj.cover_image), 1]));
                % set the seed
                rng(obj.internal_key);
                % construct a random pattern
                random_pattern = (round(rand(size(lsb)))==1);
                % place the secret in the beginning of the lsb vector
                lsb(1:obj.internal_key) = secret_bits;
                % encode it with the random pattern
                randomized_lsb = xor(lsb, random_pattern);
                % reshape lsb into the size of cover image so we can
                % replace LSB(cover_image) with our constructed lsb
                % containing the secret message.
                randomized_lsb = reshape(randomized_lsb, size(obj.cover_image));
                % LSB replacement
                obj.stego_image = bitset(obj.cover_image, 1, randomized_lsb);
            end
        end
```

همانطور که مشاهده می شود، فایل secret که قرار است درون cover_image قرار بگیرد ابتدا به صورت bit-level خوانده می شود و از تعداد bit های این فایل به عنوان کلید استفاده می کنیم که این کلید علاوه بر آن که هنگام extraction به ما نشان می دهد که چند bit از LSB تصویر stego را باید بخوانیم، به عنوان random seed نیز استفاده خواهد شد تا الگوی xor شده در مرحله ی embedding عیناً تولید شود. نکته قابل توجه در این تابع این است که مقدار LSB ابتدا از یک بردار تمام-صفر شروع می شود و سپس secret در ابتدای این بردار قرار می گیرد و بعد از آن این بردار با یک بردار تصادفی xor می شود و نهایتاً به اندازه ی تصویر cover_image، reshape می شود تا بتوانیم با استفاده از bitset مقدار LSB تصویر cover_image را تنظیم کنیم.

StegHide::ExtractSecret:

```
function extracted = ExtractSecret(obj, save_to)
            % Extracts secret from an stego image
            % extract lsb from stego_image
            lsb = (bitget(obj.stego_image, 1)==1);
            % reshape it into a vector
            lsb = uint8(reshape(lsb, [numel(obj.stego_image), 1]));
            % set the seed again
            rng(obj.internal_key);
            % reconstruct the same random pattern
            random_pattern = round(rand(size(lsb)));
            % xor twice to decode
            decoded_lsb = xor(lsb,random_pattern);
            % the key is the length of the secret message, read that much
            % from the decoded lsb
            message = decoded_lsb(1:obj.internal_key);
            % save it to a file in bit-level
            write = fopen(save_to, 'w');
            fwrite(write, message, '*ubit1');
            fclose(write);
            % return the imread of the saved file as output
            extracted = imread(save_to);
        end
```

این تابع با داشتن کلید و محلی برای ذخیره سازی فایل embed شده، آن را از تصویر stego استخراج می کند. تمام عملیات دقیقاً برعکس کاری است که در حین embedding صورت می پذیرد.