

## A Ek A: Kaynak Kod

```
1 %main
2 clear; clc; close all;
3
4 se_size = 3;
5 T = 0.35;
6 key = 4869;
7 cover = imread(['cover_ebru.jpg']);
8 secret = imread(['QR_secret_small.png']);
9 [stego, meta] = encode_adaptive_multibit(cover, secret, se_size, T, key)
10 ;
11 imwrite(stego, 'stego_key.png');
12 save('meta_key.mat', '-struct', 'meta');
13
14 recovered = decode_adaptive_multibit('stego_key.png', 'meta_key.mat',
    key);
```

Kod 1: Uyarlamalı kenar tabanlı çok bitli LSB steganografi yönteminin gerçekleştiren ana MATLAB betiği.

```
1 function [stego, meta] = encode_adaptive_multibit(COVER, SECRET,
    se_size, T, key, payload_limit)
2 % Adaptif bit-şabana-piksel + anahtar ıtabanlı şrastgeleletirme ile
    uyarlanabilir kenar ıtabanlı LSB gömme
3 % - ızayf kenar pikselleri: 1 bit (LSB)
4 % - güçlü kenar pikselleri: 2 bit (LSB + 2. LSB)
5
6 % ŞGiri parametreleri kontrolü (ıVarsaylan ııısnrsz yük)
7 if nargin < 6
8     payload_limit = inf;
9 end
10
11 % Görüntüleri ğbellee alma ve veri tipini ıhazrlama
12 cover = COVER;
13 secret = SECRET;
14 secret = uint8(secret);
15
16 % Kapak görüntüsünün RGB ıformatnda ğolduunu ğdorula
17 if size(cover,3) ~= 3
18     error('Kapak görüntüsü RGB (H×W×3) ıformatnda ııolmaldr.');
```

```

23 gray_cover = rgb2gray(cover);
24 % Canny 1algoritmas ile temel 1kenarlar tespit et
25 edges = edge(gray_cover, 'canny');
26
27 % Morfolojik 1Geniletme (Dilation) 1ilemi
28 % Belirlenen 1kenarlar SE (1yapsal eleman) kadar 1genileterek maske
1soluturur
29 se = strel('square', se_size);
30 mask = imdilate(edges, se);
31
32 % --- 1ANALZ VE SINIFLANDIRMA ---
33 % Sobel gradyan hesaplama (Kenar 1iddetini 1l1mek i1in)
34 G = imgradient(gray_cover, 'sobel');
35 G = mat2gray(G); % Gradyan 11deerlerini [0,1] 111aralna normalize et
36
37 % G1111 1kenarlar belirle (Maske i1inde ve 1eik 11deerleri T'den b111k
olanlar)
38 strong = mask & (G > T);
39
40 % bits_map (Bit 1Haritas): 0 = G11me yok, 1 = 1Zayf kenar, 2 = G11
11 kenar
41 bits_map = zeros(size(mask), 'uint8');
42 bits_map(mask) = 1; % T1m maske 11alann 1nce 1zayf kabul et
43 bits_map(strong) = 2; % G1111 1gradyanl 1alanlar g1ncelle
44
45 % Gizli g1r1nt111 1D bit dizisine (bitstream) d1n111t1r
46 secret_bits = int2bit(secret(:), 8);
47 secret_bits = secret_bits(:)';
48
49 % 11111 payload (y1k) 111111 1belirtilmisse bit dizisini 1k1p
50 if payload_limit < length(secret_bits)
51     secret_bits = secret_bits(1:payload_limit);
52 end
53
54 % Kapasite kontrol1: Mevcut alan gizli veriyi 11tayabilir mi?
55 total_bits = numel(secret_bits);
56 capacity_bits = sum(bits_map(:));
57 if total_bits > capacity_bits
58     error('G11me 111baarsz: Gizli veri (%d bit) kapasiteyi (%d bit)
11ayor.', ...
59         total_bits, capacity_bits);
60 end
61
62 % --- ANAHTAR TABANLI 11RASTGELELETRME (G11VENLK) ---
63 % Veri g1m1lebilecek piksellerin 11dorusal indekslerini bul
64 eligible_idx = find(bits_map(:) > 0);

```

```

65 % Kullanıcı anahtar ile rastgele sayı üreticini şifalat (Aynı
anahtar çözmek için şarttır)
66 rng(key, 'twister');
67 % Piksel sırasını ışıkarır
68 perm = randperm(numel(eligible_idx));
69 eligible_idx = eligible_idx(perm);
70
71 % --- İVER GÖMME (EMBEDDING) SÜRECİ ---
72 stego = cover;
73 bit_idx = 1;
74 for k = 1:numel(eligible_idx)
75     if bit_idx > total_bits
76         break; % Tüm veriler gömüldüyse döngüyü sonlandır
77     end
78
79     idx = eligible_idx(k); % ışıkarılmış dorusal indeks
80     b = bits_map(idx); % Mevcut pikselin kapasitesi (1
veya 2 bit)
81     [i, j] = ind2sub(size(bits_map), idx); % Matris koordinatlarını
bul
82
83     if b == 1
84         % Zayıf kenar: Sadece Mavi kanalın 1. LSB'sine göm
85         stego(i,j,3) = bitset(stego(i,j,3), 1, secret_bits(bit_idx)
);
86         bit_idx = bit_idx + 1;
87     elseif b == 2
88         % Güçlü kenar: Önce 1. LSB'ye göm
89         stego(i,j,3) = bitset(stego(i,j,3), 1, secret_bits(bit_idx)
);
90         bit_idx = bit_idx + 1;
91         % Eğer hala bit varsa, 2. LSB'ye de göm (Çoklu bit özelliği)
92         if bit_idx <= total_bits
93             stego(i,j,3) = bitset(stego(i,j,3), 2, secret_bits(
bit_idx));
94             bit_idx = bit_idx + 1;
95         end
96     end
97 end
98
99 payload_bits = bit_idx - 1;
100
101 % --- İKALTE ÖLÇÜMLER VE METADATA ---
102 % PSNR ve MSE hesaplama (Görüntü kalitesi analizi)
103 [psnr_val, mse_val] = psnr(stego, cover);

```

```

104 % SSIM hesaplama (ıYapsal benzerlik analizi)
105 ssim_val = ssim(stego, cover);
106
107 % Meta verileri ıyap (struct) olarak kaydet (Çözücü için gereklidir
108 )
109 meta.mask = mask;
110 meta.bits_map = bits_map;
111 meta.payload_bits = payload_bits;
112 meta.secret_size = size(secret);
113 meta.secret_class = class(secret);
114 meta.se_size = se_size;
115 meta.T = T;
116
117 % Sonuçları Görselleştirme
118 figure('Name','Orişinal vs Stego Görüntü','NumberTitle','off');
119 tiledlayout(1,2,'Padding','compact','TileSpacing','compact');
120 nexttile; imshow(cover); title('Orişinal Kapak Resmi');
121 nexttile; imshow(stego); title('Stego Görüntüsü (Veri şGizlenmi)');
122 end

```

Kod 2: Uyarlanabilir çok bitli LSB veri gömme (encoder) algoritmasının MATLAB implementasyonu.

```

1 function extracted_image = decode_adaptive_multibit(stego_path,
2 meta_path, key)
3 % Meta veri + anahtar ıtabanlı şrastgeleletirme kullanarak
4 % uyarlanabilir çok bitli stego görüntüden gizli görüntüyü çözer
5
6 % ıDosyaları oku
7 stego = imread(stego_path);
8 meta = load(meta_path); % Meta verilerin ıyap (struct) olarak yü
9 klenmesi beklenir
10
11 bits_map = meta.bits_map;
12 payload_bits = meta.payload_bits;
13 [rows, cols, ~] = size(stego);
14
15 % Boyut kontrolü: Bit ıharitas ile stego görüntüsü uyumlu mu?
16 if ~isequal(size(bits_map), [rows, cols])
17     error('Boyut şuyumsuzluu: bits_map ve stego ıboyutlar şşelemiyor
18     .');
19 end
20
21 % --- AYNI ıanahtar kullanarak AYNI rastgele ıısray şçözümler ---
22 % Veri gömülmüş olan piksellerin lineer indekslerini bul
23 eligible_idx = find(bits_map(:) > 0);

```

```

21
22 % Rastgele ısay üreticini (RNG) anahtar ile şbalat (ııKodlayc ile
ıayn ıolmal)
23 rng(key, 'twister');
24 perm = randperm(numel(eligible_idx));
25 eligible_idx = eligible_idx(perm); % İndeksleri ııkartr
26
27 % Çıkarılacak bitler için yer ıayr
28 extracted_bits = zeros(1, payload_bits, 'uint8');
29 bit_idx = 1;
30
31 % ıııııKartrıl indeksler üzerinde döngü şbalat
32 for k = 1:numel(eligible_idx)
33     if bit_idx > payload_bits
34         break; % Tüm veriler ııkarıldıysa döngüden çık
35     end
36
37     idx = eligible_idx(k);
38     b = bits_map(idx); % Bu pikselde kaç bit gizli ğolduunu kontrol
    et (1 veya 2)
39     [i, j] = ind2sub(size(bits_map), idx); % Lineer indeksi
    koordinatlara çevir
40
41     if b == 1
42         % Mavi ıkanalın 1. LSB bitini oku
43         extracted_bits(bit_idx) = bitget(stego(i,j,3), 1);
44         bit_idx = bit_idx + 1;
45     elseif b == 2
46         % Mavi ıkanalın 1. LSB bitini oku
47         extracted_bits(bit_idx) = bitget(stego(i,j,3), 1);
48         bit_idx = bit_idx + 1;
49
50         % ğEer hala ııkarılacak bit varsa 2. LSB bitini oku
51         if bit_idx <= payload_bits
52             extracted_bits(bit_idx) = bitget(stego(i,j,3), 2);
53             bit_idx = bit_idx + 1;
54         end
55     end
56 end
57
58 % Bit dizisini (bitstream) tekrar byte (tam ısay) ıformatna dönüştür
59 extracted_bytes = bit2int(reshape(extracted_bits, 8, []), 8);
60
61 % Görüntüyü orijinal ıboyutlarına göre yeniden ııyaplandır
62 extracted_image = reshape(uint8(extracted_bytes), meta.secret_size)
;

```

```

63
64     % Veri tipini orijinal haline getir (uint8 vb.)
65     extracted_image = cast(extracted_image, meta.secret_class);
66
67     % Sonucu görselletir
68     figure; imshow(extracted_image);
69     title('1Kurtarlan Gizli Görüntü (Anahtar-şrastgeleletirmeli Adaptif
        Çok-Bitli)');
70 end

```

Kod 3: Uyarlamalı çok bitli LSB veri çıkarma (decoder) algoritmasının MATLAB implementasyonu.

```

1      clear; clc; close all;
2
3      % --- PARAMETRELER ---
4      se_size = 3;          % şGenileme (Dilation) için ıyapsal eleman boyutu
5      T = 0.35;            % Güçlü/ıZayf kenar ııayrm için şeik şdeeri
6      key = 4869;          % ışıKartrma için güvenlik ıanahtar
7      secret = imread('QR_secret.png'); % Tüm deneylerde ıkullanılacak gizli
        veri (QR kod)
8
9      % Test edilecek kapak resimleri listesi
10     covers = {
11         'cover_btu.jpeg'
12         'cover_dessert.jpeg'
13         'cover_ebru.jpg'
14     };
15     num_covers = length(covers);
16
17     %% --- İİNMUM İKAPASTE HESAPLAMA ---
18     % Tüm resimler ıarasnda en küçük kapasiteye sahip ıolan bulup deney A ı
        çin sabit yük belirleriz.
19     capacities = zeros(num_covers,1);
20     for k = 1:num_covers
21         cover = imread(covers{k});
22         gray = rgb2gray(cover);
23
24         % Kenar ve gradyan analizi
25         edges = edge(gray,'canny');
26         se = strel('square', se_size);
27         mask = imdilate(edges, se);
28         G = imgradient(gray,'sobel');
29         G = mat2gray(G);
30
31         % Bit ıharitas şolurma

```



```

74     KL(k) = sum(h_cover .* log((h_cover + eps) ./ (h_stego + eps)));
75     ImageName(k) = covers{k};
76 end
77
78 Results_A = table(ImageName, PSNR_A, SSIM_A, L1, L2, CHI, KL);
79 disp('--- Deney A Sonuçları (Sabit Yük) ---');
80 disp(Results_A);
81
82 %% --- DENEY B: İKAPASTEYE GÖRE İİUYARLANABLR YÜK ---
83 % Bu bölümde her resmin kendi maksimum kapasitesinin %80'i kadar veri g
    ömülür.
84 Capacity_B = zeros(num_covers,1);
85 Payload_B = zeros(num_covers,1);
86 PSNR_B = zeros(num_covers,1);
87 SSIM_B = zeros(num_covers,1);
88
89 for k = 1:num_covers
90     cover = imread(covers{k});
91     gray = rgb2gray(cover);
92
93     % Kapasite tahmini (Yeniden ıhesaplanır çünkü her resmin dokusu
    ııfarklıdır)
94     edges = edge(gray, 'canny');
95     se = strel('square', se_size);
96     mask = imdilate(edges, se);
97     G = imgradient(gray, 'sobel');
98     G = mat2gray(G);
99
100     strong = mask & (G > T);
101     bits_map = zeros(size(mask), 'uint8');
102     bits_map(mask) = 1;
103     bits_map(strong) = 2;
104
105     capacity = sum(bits_map(:));
106     payload = floor(0.8 * capacity); % Resme özel %80 doluluk ıoran
107
108     % Gömme İşlemi
109     [stego, meta] = encode_adaptive_multibit(cover, secret, se_size, T,
        key, payload);
110
111     Capacity_B(k) = capacity;
112     Payload_B(k) = payload;
113     PSNR_B(k) = psnr(stego, cover);
114     SSIM_B(k) = ssim(stego, cover);
115     ImageName(k) = covers{k};
116 end

```

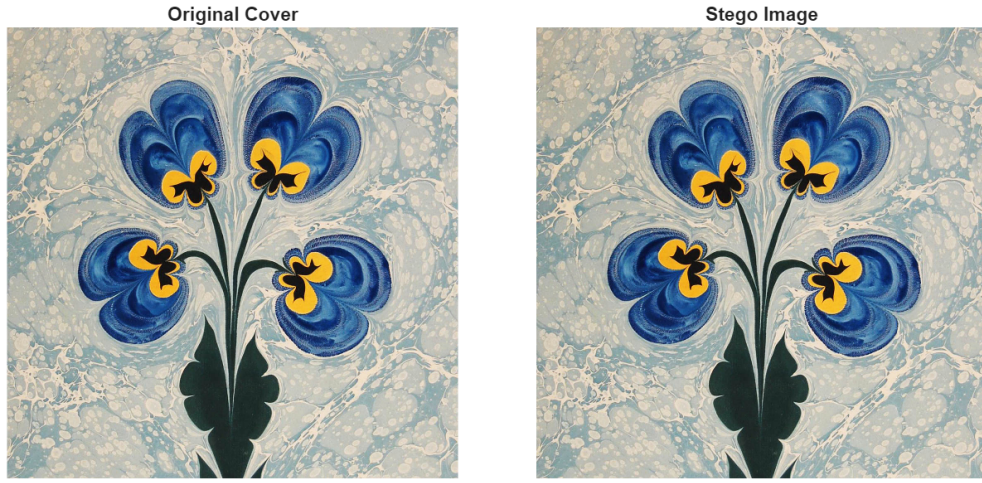
```
117  
118 Results_B = table(ImageName, Capacity_B, Payload_B, PSNR_B, SSIM_B);  
119 disp('--- Deney B Sonuçları (Resme Özel görsel Yük) ---');  
120 disp(Results_B);
```

Kod 4: Uyarlamalı kenar tabanlı çok bitli LSB steganografi yönteminin sabit ve kapasite-uyarlamalı yük senaryoları altında deneysel değerlendirilmesini gerçekleştiren ana MATLAB betiği.

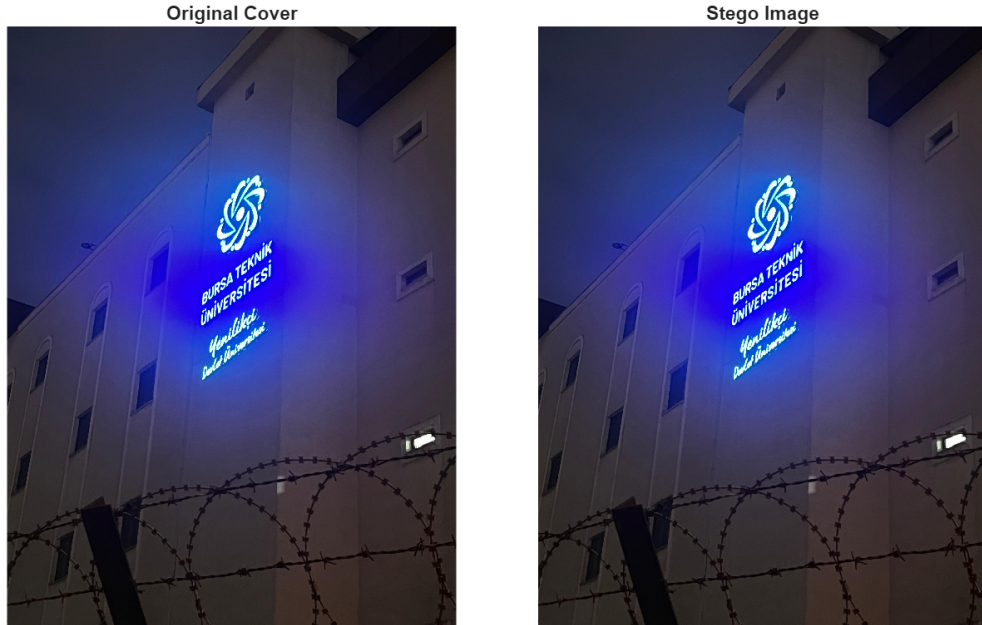
## B Ek şekiller



Şekil 1: cover\_dessert.jpg: kapak görüntüsü için kapak görüntüsü (sol), stego görüntüsü (sağ)



Şekil 2: cover\_ebru.jpg: kapak görüntüsü için kapak görüntüsü (sol), stego görüntüsü (sağ)



Şekil 3: cover\_btu.jpg: kapak görüntüsü için kapak görüntüsü (sol), stego görüntüsü (sağ)