

## 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,
2     se_size, T, key, payload_limit)
% Adaptif bit-şıbana-piksel + anahtar ıtabanlı şrastgeleştirme ile
% uyarlanabilir kenar ıtabanlı LSB gömme
% - ızayf kenar pikselleri: 1 bit (LSB)
% - güclü kenar pikselleri: 2 bit (LSB + 2. LSB)
%
% Giriş parametreleri kontrolü (Varsayılan ııısnrsz yük)
5 if nargin < 6
    payload_limit = inf;
end
%
% Görüntüleri ıgbellee alma ve veri tipini ıhazrlama
11 cover = COVER;
12 secret = SECRET;
13 secret = uint8(secret);
%
16 % Kapak görüntüsünün RGB ıformatnda ıoldunu ıdorula
17 if size(cover,3) ~= 3
    error('Kapak görüntüsü RGB (HxWx3) ıformatnda ııolmalıdır.');
end
%
21 % --- MASKE ıSOLUTURMA SÜREC ---
22 % Renkli görüntüyü kenar ıalglama için gri tonlamaya dönüştür
```

```

23 gray_cover = rgb2gray(cover);
24 % Canny algoritmas ile temel ikenarlar tespit et
25 edges = edge(gray_cover, 'canny');
26
27 % Morfolojik şenileme (Dilation) şilemi
28 % Belirlenen ikenarlar SE (iyapsal eleman) kadar şenileterek maske
29 % soluturur
30 se = strel('square', se_size);
31 mask = imdilate(edges, se);
32
33 % --- İANALZ VE SINIFLANDIRMA ---
34 % Sobel gradyan hesaplama (Kenar şiddetini ölçmek için)
35 G = imgradient(gray_cover, 'sobel');
36 G = mat2gray(G); % Gradyan değerlerini [0,1] aralına normalize et
37
38 % Güçlü ikenarlar belirle (Maske içinde ve şeik değeri T'den büyük
39 % olanlar)
40 strong = mask & (G > T);
41
42 % bits_map (Bit Haritası): 0 = Gömme yok, 1 = Zayıf kenar, 2 = Güçlü
43 % kenar
44 bits_map = zeros(size(mask), 'uint8');
45 bits_map(mask) = 1; % Tüm maske alan önce zayıf kabul et
46 bits_map(strong) = 2; % Güçlü gradyan alanlar güncelle
47
48 % Gizli görüntüyü 1D bit dizisine (bitstream) dönüştür
49 secret_bits = int2bit(secret(:), 8);
50 secret_bits = secret_bits(:)';
51
52 % Eer payload (yük) ısnr şbelirtilmise bit dizisini krp
53 if payload_limit < length(secret_bits)
54     secret_bits = secret_bits(1:payload_limit);
55 end
56
57 % Kapasite kontrolü: Mevcut alan gizli veriyi şıtayabilir mi?
58 total_bits = numel(secret_bits);
59 capacity_bits = sum(bits_map(:));
60 if total_bits > capacity_bits
61     error('Gömme şıtayabilir: Gizli veri (%d bit) kapasiteyi (%d bit)
62     %sayor.', ...
63         total_bits, capacity_bits);
64 end
65
66 % --- ANAHTAR TABANLI ŞİRASTGELELETRME (GÜVENLİK) ---
67 % Veri gömebilecek piksellerin dorusal indekslerini bul
68 eligible_idx = find(bits_map(:) > 0);

```

```

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

```

```

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

```

Kod 2: Uyarlamalı 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 görüntüsünden gizli görüntüyü çözer
5
6 % ıDosyalar 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 ıguyumsuzluu: bits_map ve stego ıboyutlar şslemeiyor
18 .');
19 end
20
21 % --- AYNI ıanahtar kullanarak AYNI rastgele ıısrıy şolutur ---
22 % Veri gömülüş olan piksellerin lineer indekslerini bul
23 eligible_idx = find(bits_map(:) > 0);

```

```

21
22 % Rastgele ısay üreteçini (RNG) anahtar ile şbalat (ııKodlayc ile
23 ıayn ıolmal)
24 rng(key, 'twister');
25 perm = randperm(numel(eligible_idx));
26 eligible_idx = eligible_idx(perm); % İndeksleri ışıkartr
27
28 % Çıkaracak bitler için yer ıayr
29 extracted_bits = zeros(1, payload_bits, 'uint8');
30 bit_idx = 1;
31
32 % ışılışKartrlm indeksler üzerinde döngü şbalat
33 for k = 1:numel(eligible_idx)
34     if bit_idx > payload_bits
35         break; % Tüm veriler ıııkarldysa döngüden çık
36     end
37
38     idx = eligible_idx(k);
39     b = bits_map(idx); % Bu pikselde kaç bit gizli ıoldunu kontrol
40     et (1 veya 2)
41     [i, j] = ind2sub(size(bits_map), idx); % Lineer indeksi
42     koordinatlara ıevir
43
44     if b == 1
45         % Mavi ıkanaln 1. LSB bitini oku
46         extracted_bits(bit_idx) = bitget(stego(i,j,3), 1);
47         bit_idx = bit_idx + 1;
48     elseif b == 2
49         % Mavi ıkanaln 1. LSB bitini oku
50         extracted_bits(bit_idx) = bitget(stego(i,j,3), 1);
51         bit_idx = bit_idx + 1;
52
53         % ıEer hala ıııkaracak bit varsa 2. LSB bitini oku
54         if bit_idx <= payload_bits
55             extracted_bits(bit_idx) = bitget(stego(i,j,3), 2);
56             bit_idx = bit_idx + 1;
57         end
58     end
59 end
60
61 % Bit dizisini (bitstream) tekrar byte (tam ısay) ıformatna dönüştür
62 extracted_bytes = bit2int(reshape(extracted_bits, 8, []), 8);
63
64 % Görüntüyü orijinal ıboyutlarna göre yeniden ııyaplandır
65 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 görselleştir
68    figure; imshow(extracted_image);
69    title('Kurtarlan Gizli Görüntü (Anahtar-şartlı Adaptif
70    Çok-Bitli)');
71 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;           % Genileştirme (Dilation) için yapısal eleman boyutu
5 T = 0.35;              % Güçlü/zayıf kenar tıayrm için şeik şdeeri
6 key = 4869;            % ışıKartrma için güvenlik anahtar
7 secret = imread('QR_secret.png'); % Tüm deneylerde kullanılacak gizli
8     veri (QR kod)
9
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 %% --- İNMUM İKAPASTA HESAPLAMA ---
18 % Tüm resimler arasında en küçük kapasiteye sahip olan bulup deney A i
19 %n sabit yük belirleriz.
20 capacities = zeros(num_covers,1);
21 for k = 1:num_covers
22     cover = imread(covers{k});
23     gray = rgb2gray(cover);
24
25     % Kenar ve gradyan analizi
26     edges = edge(gray, 'canny');
27     se = strel('square', se_size);
28     mask = imdilate(edges, se);
29     G = imgradient(gray, 'sobel');
30     G = mat2gray(G);
31
32     % Bit ıharitas şoluturma

```

```

32     strong = mask & (G > T);
33     bits_map = zeros(size(mask), 'uint8');
34     bits_map(mask) = 1; % 1Zayf kenarlar
35     bits_map(strong) = 2; % Güçlü kenarlar
36     capacities(k) = sum(bits_map(:)); % Toplam bit kapasitesi
37 end
38
39 % Deney A için yük: En küçük kapasitenin %80'i kadar sabit veri gömülü lecek.
40 payload_fixed = floor(0.8 * min(capacities));
41 fprintf('Deney A Sabit Yük: %d bit\n', payload_fixed);
42
43 %% --- DENEY A: İSABT YÜK İİANALZ (Görüntü Kalitesi şışılıkarlatmas) ---
44 % Bu bölümde ifarklı resimlere AYNI miktarda veri gömülüğünde sonuçlar şışılıkarlatrır.
45 ImageName = strings(num_covers,1);
46 PSNR_A = zeros(num_covers,1);
47 SSIM_A = zeros(num_covers,1);
48 L1 = zeros(num_covers,1); % Histogram ifark metrikleri
49 L2 = zeros(num_covers,1);
50 CHI = zeros(num_covers,1); % Chi-Square testi
51 KL = zeros(num_covers,1); % Kullback-Leibler Divergence İ(statistiksel benzerlik)
52
53 for k = 1:num_covers
54     cover = imread(covers{k});
55     [stego, meta] = encode_adaptive_multibit(cover, secret, se_size, T, key, payload_fixed);
56
57     % Görüntü Kalite Metrikleri
58     PSNR_A(k) = psnr(stego, cover);
59     SSIM_A(k) = ssim(stego, cover);
60
61     % --- İHSTOGRAM İİANALZ İ(statistiksel Güvenlik) ---
62     % Mavi kanaldaki (veri gömülüen kanal) histogram şışdeiimini ölçer
63     h_cover = imhist(cover(:,:,3));
64     h_stego = imhist(stego(:,:,3));
65
66     % Normalize etme (ııolaslk şıyounluk fonksiyonuna çevirme)
67     h_cover = h_cover / sum(h_cover);
68     h_stego = h_stego / sum(h_stego);
69
70     % Histogramlar ıarasndaki ifarklarn ıhesaplanması
71     L1(k) = sum(abs(h_cover - h_stego));
72     L2(k) = norm(h_cover - h_stego);
73     CHI(k) = sum((h_cover - h_stego).^2 ./ (h_cover + eps));

```

```

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 Sonuçlar (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
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 hesaplanır çünkü her resmin dokusu
94     % farklıdır)
95     edges = edge(gray, 'canny');
96     se = strel('square', se_size);
97     mask = imdilate(edges, se);
98     G = imgradient(gray, 'sobel');
99     G = mat2gray(G);
100
101    strong = mask & (G > T);
102    bits_map = zeros(size(mask), 'uint8');
103    bits_map(mask) = 1;
104    bits_map(strong) = 2;
105
106    capacity = sum(bits_map(:));
107    payload = floor(0.8 * capacity); % Resme özel %80 doluluk oranı
108
109    % Gömme işlemi
110    [stego, meta] = encode_adaptive_multibit(cover, secret, se_size, T,
111                                              key, payload);
112
113    Capacity_B(k) = capacity;
114    Payload_B(k) = payload;
115    PSNR_B(k) = psnr(stego, cover);
116    SSIM_B(k) = ssim(stego, cover);
117    ImageName(k) = covers{k};
118 end

```

```

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

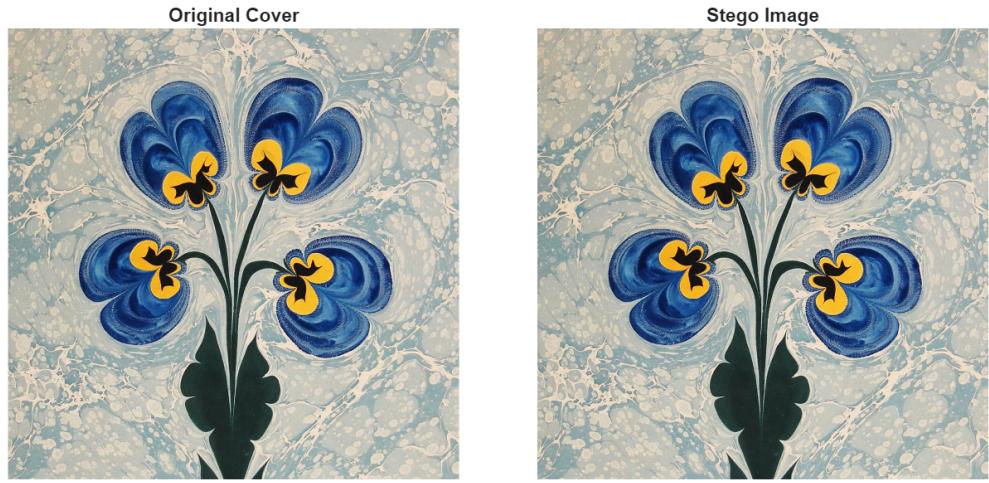
```

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

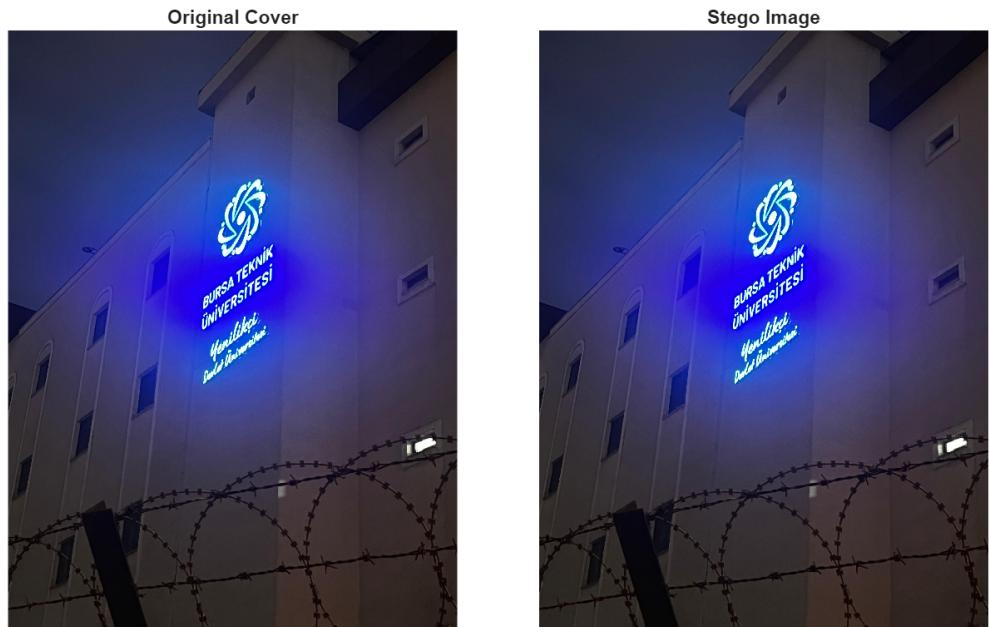
## B Ek şekiller



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)