Final DIP Project

- Pratik Puri Goswami(16110121)
- Vasu Bhalothia (16110174)

Assigned TA: Vinay Verma

PAPERS Evaluated-

1) Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis

2) Improved DCT coefficient analysis for forgery localization in JPEG images

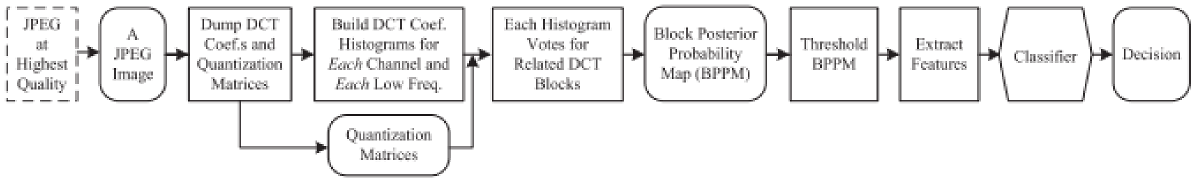
Introduction

There has been a long history of image forgery. In the early days, dark-room skills were used to print multiple fragments of photos onto a single photograph paper. In the current digital era, image/video forgery becomes much easier. The techniques involve naive cutting and pasting, matting for perfect blending texture synthesis for synthesizing new contents.
Image forensic technologies can be categorized as active ones and passive ones.
Active image forensic methods mainly insert digital watermark to images/videos at the instant of their acquisition. The integrity of images/videos can be checked by detecting the change in the watermark.
In contrast, passive image forensic aims at developing technologies for tampered image/video detection without using knowledge beyond the image/video itself.

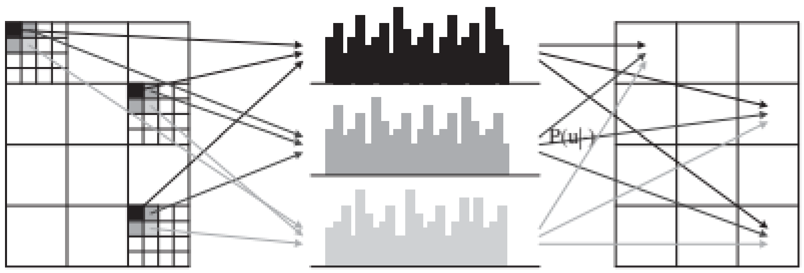
In the first paper, a fast and fully automatic detection method for JPEG images is proposed. The reason we target JPEG images is because JPEG is the most widely used image format. Particularly in digital cameras, JPEG may be the most preferred image format due to its efficiency of storage. Our method is based on the DQ effect. Intuitively speaking, the DQ effect is the exhibition of periodic peaks and valleys in the histograms of the discrete cosine transform, DCT, coefficients.)

In the second paper, a proposal using a statistical test to discriminate between original and forged regions in JPEG images is made under the hypothesis that the former are doubly compressed while the latter are singly compressed. New probability models for the DCT coefficients of singly and doubly compressed regions are proposed, together with a reliable method for estimating the primary quantization factor in the case of double compression.

Basic outline of the Algorithm for the first paper


JPEG at Highest Quality
A JPEG Image
Dump DCT Coef.s and Quantization Matrices
Build DCT Coef. Histograms for Each Channel and Each Low Freq.
Each Histogram Votes for Related DCT Blocks
Block Posterior Probability Map (BPPM)
Threshold BPPM
Extract Features
Classifier
Decision
Quantization Matrices

Given a JPEG image, we first dump its DCT coefficients and quantization matrices for YUV channels. If the image is originally stored in other lossless format, we first convert it to the JPEG format at the highest compression quality. Then we build histograms for each channel and each frequency. Note that the DCT coefficients are of 64 frequencies in total, varying from (0,0) to (7,7). For each frequency, the DCT coefficients of all the blocks can be gathered to build a histogram. Moreover, a color image is always converted into the YUV space for JPEG compression. Therefore, we can build at most 64 × 3 = 192 histograms of DCT coefficients of different frequencies and different channels.However, as high frequency DCT coefficients are often quantized to zeros, only the histograms of low frequencies of each channel are useful. For each block in the image, using one histogram we can compute one probability of it being a tampered block, by checking the DQ effect of this histogram. With all the available histograms, we can accumulate the probabilities to give the posterior probability of this block being unchanged . Then the block posterior probability map (BPPM)is thresholded to differentiate the possibly tampered region and possibly unchanged region. With such a segmentation, a four-dimensional feature vector is computed for the image. Finally, a trained SVM is applied to decide whether the image is tampered. If it is tampered, then the segmented tampered region is also output.



Basic outline for the Algorithm for the second paper

DCT coefficients of unmodified areas will undergo a double JPEG compression thus exhibiting double quantization (DQ) artifacts, while DCT coefficients of forged areas will result from a single compression and will likely present no artifacts. In the following, we will refer to this scenario as the single compression forgery (SCF) hypothesis.The idea is to use Bayesian inference to assign to each DCT coefficient a probability of being doubly quantized. Such probabilities,accumulated over each 8 × 8 block, will provide a DQ probability map allowing us to tell original areas (high DQ probability) from tampered areas (low DQ probability).Bayesian inference is based on the probability distribution of DCT coefficients conditional to the hypothesis of being tampered, i.e., p(x|H0), where x is the value of the DCT coefficient and H0 (H1) indicates the hypothesis of being tampered (original).

Background

JPEG compression

The compression of JPEG images involves three basic steps:
(1) DCT: An image is first divided into DCT blocks. Each block is subtracted by 128 and transformed to the YUV color space. Finally DCT is applied to each channel of the block.
(2) Quantization: the DCT coefficients are divided by a quantization step and rounded to the nearest integer.
(3) Entropy coding: lossless entropy coding of quantized DCT coefficients..
The quantization steps for different frequencies are stored in quantization matrices (luminance matrix for Y channel or chroma matrix for U and V channels). Two important points to be noted are-
(1) The higher the compression quality is, the smaller the quantization step will be, and vice versa.
(2) The quantization step may be different for different frequencies and different channels.

The decoding of a JPEG image involves the inverse of the previous three steps taken in reverse order: entropy decoding, de-quantization, and inverse DCT (IDCT). Unlike the other two operations, the quantization step is not invertible.

Consequently, when an image is doubly JPEG-compressed, it will undergo the following steps and the DCT coefficients will change accordingly:
(1) The first compression:
(a) DCT (suppose after this step a coefficient value u is obtained).
(b) The first quantization with quantization step Q1.
(2) The first decompression:
(a) Dequantization with Q1 .
(b) IDCT.
(3) The second compression:
(a) DCT.
(b) The second quantization with quantization step Q2.

DQ effect

DQ effect

After the Second compression we get;

$$Q_{q_1 q_2}(u_1) = \left[\left[\frac{u}{q_1} \right] \frac{q_1}{q_2} \right] = u_2$$

where q_1 and q_2 are the first ~~qom~~ quantization step and second quantization step respectively.

$$u_2 - \frac{1}{2} \leq \left[\frac{u_1}{q_1} \right] \frac{q_1}{q_2} < u_2 + \frac{1}{2}$$ {defination of rounding operation}

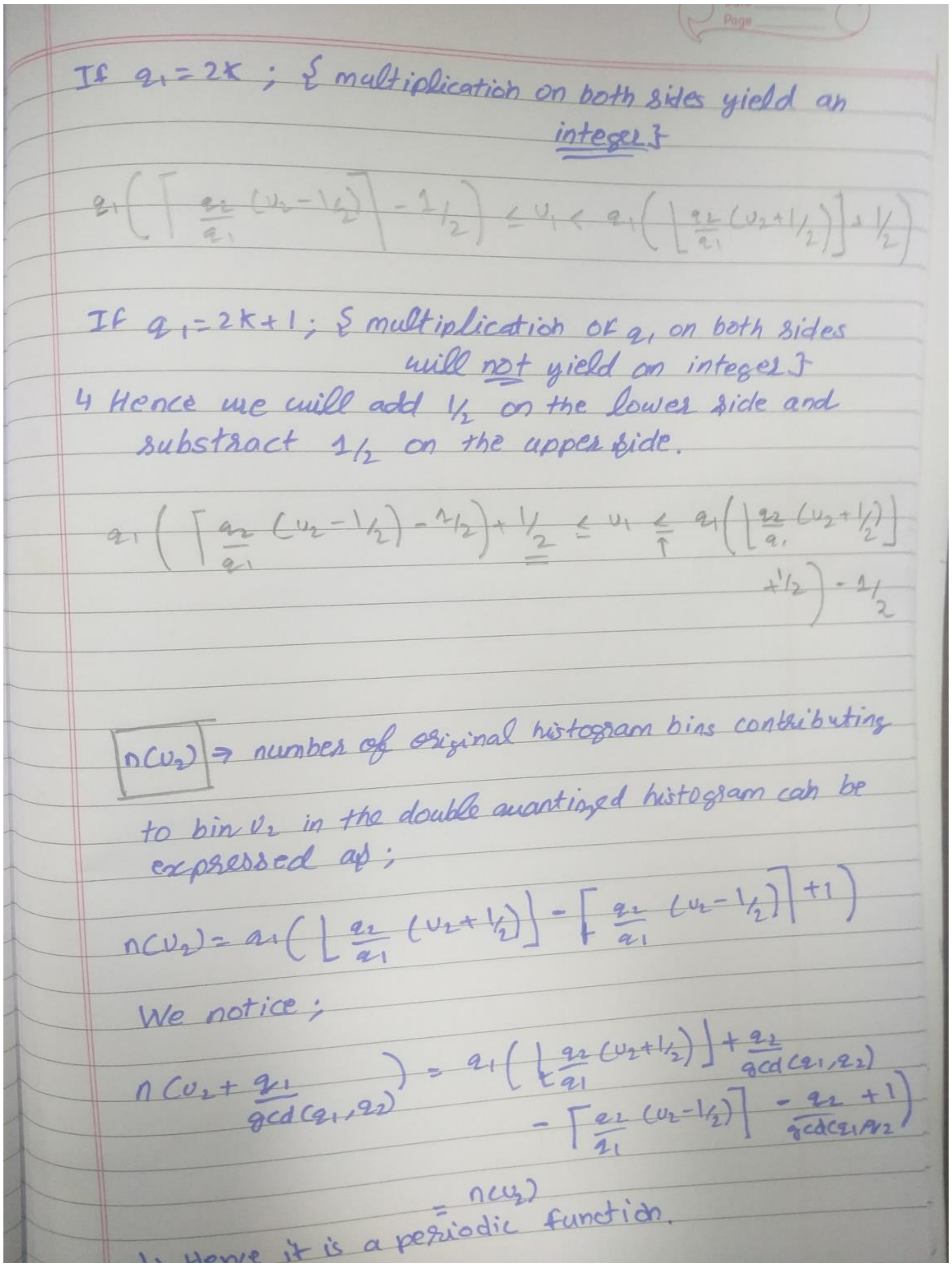
↳ Now $\frac{q_2}{q_1}$ might not be an integer. For Excample it might be 6.5. Next nearest integer is 7. Hence we need to ceil on the lower side.
Previous largest integer is 6. Hence we need to floor on the upper side.
Hence we arrive at;

$$\left\lceil \left(u_2 - \frac{1}{2}\right) \frac{q_2}{q_1} \right\rceil \leq \left[\frac{u_1}{q_1} \right] \leq \left\lfloor \frac{q_2}{q_1} \left(u_2 + \frac{1}{2}\right) \right\rfloor$$

Now it becomes;

$$\left\lceil \left(u_2 - \frac{1}{2}\right) \frac{q_2}{q_1} \right\rceil - \frac{1}{2} \leq \frac{u_1}{q_1} < \left\lfloor \frac{q_2}{q_1} \left(u_2 + \frac{1}{2}\right) \right\rfloor + \frac{1}{2}$$

2 case arise;
$q_1 = 2K$ or $2K+1$



If $q_1 = 2k$; { multiplication on both sides yield an <u>integer</u> }

$$q_1\left(\left\lceil \frac{q_2}{q_1}\left(v_2-\tfrac{1}{2}\right)\right\rceil-\tfrac{1}{2}\right) \le v_1 < q_1\left(\left\lfloor \frac{q_2}{q_1}\left(v_2+\tfrac{1}{2}\right)\right\rfloor+\tfrac{1}{2}\right)$$

If $q_1 = 2k+1$; { multiplication of q_1 on both sides will <u>not</u> yield an integer }

↳ Hence we will add ½ on the lower side and substract ½ on the upper side.

$$q_1\left(\left\lceil \frac{q_2}{q_1}\left(v_2-\tfrac{1}{2}\right)-\tfrac{1}{2}\right)+\underline{\tfrac{1}{2}} \le v_1 \underset{\uparrow}{\le} q_1\left(\left\lfloor \frac{q_2}{q_1}\left(v_2+\tfrac{1}{2}\right)\right\rfloor+\tfrac{1}{2}\right)-\tfrac{1}{2}$$

$\boxed{n(v_2)}$ ⇒ number of original histogram bins contributing to bin v_2 in the double quantized histogram can be expressed as;

$$n(v_2) = q_1\left(\left\lfloor \frac{q_2}{q_1}\left(v_2+\tfrac{1}{2}\right)\right\rfloor - \left\lceil \frac{q_2}{q_1}\left(v_2-\tfrac{1}{2}\right)\right\rceil + 1\right)$$

We notice;

$$n\left(v_2+\frac{q_1}{\gcd(q_1,q_2)}\right) = q_1\left(\left\lfloor \frac{q_2}{q_1}\left(v_2+\tfrac{1}{2}\right)\right\rfloor + \frac{q_2}{\gcd(q_1,q_2)} - \left\lceil \frac{q_2}{q_1}\left(v_2-\tfrac{1}{2}\right)\right\rceil - \frac{q_2}{\gcd(q_1,q_2)} + 1\right)$$

$$= n(v_2)$$

∴ Hence it is a periodic function.

Observation:
The period is chosen as q1/gcd(q1,q2) otherwise q1/q2 is also a period but it is not an integer.

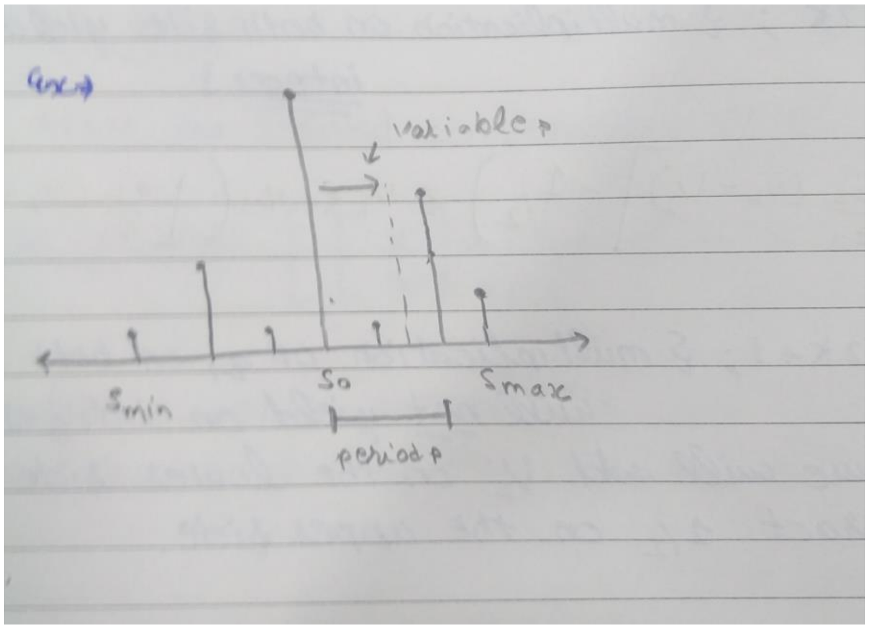
Period Estimation for paper1

Suppose s0 is the index of the bin that has the largest value. For each p between 1 and smax/20, we compute the following quantity:

$$H(p) = \frac{1}{i_{max} - i_{min} + 1} \sum_{i=i_{min}}^{i_{max}} [h(i \cdot p + s_0)]^{\alpha},$$

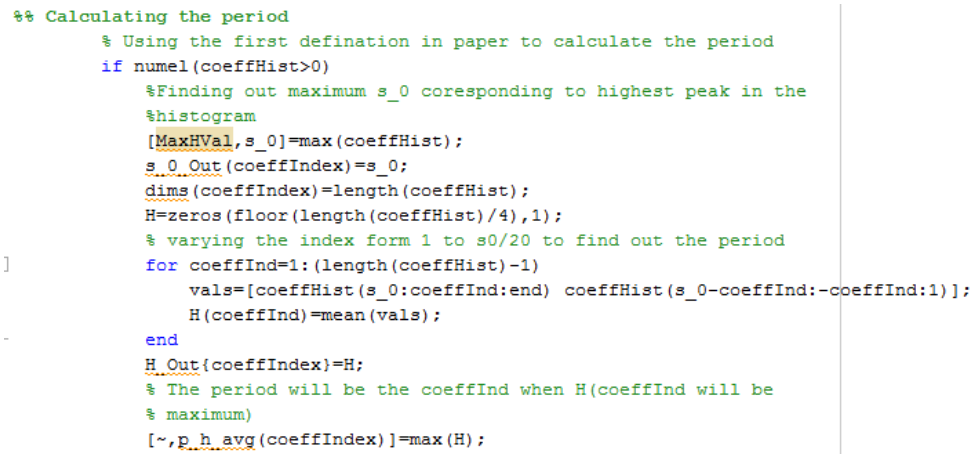
where imax = (smax - s0)/p, imin = (smin - s0)/p, smax and smin are the maximum and minimum index of the bins in the histogram, respectively, and is a parameter (can be simply chosen as 1).\

Here we consider an example :



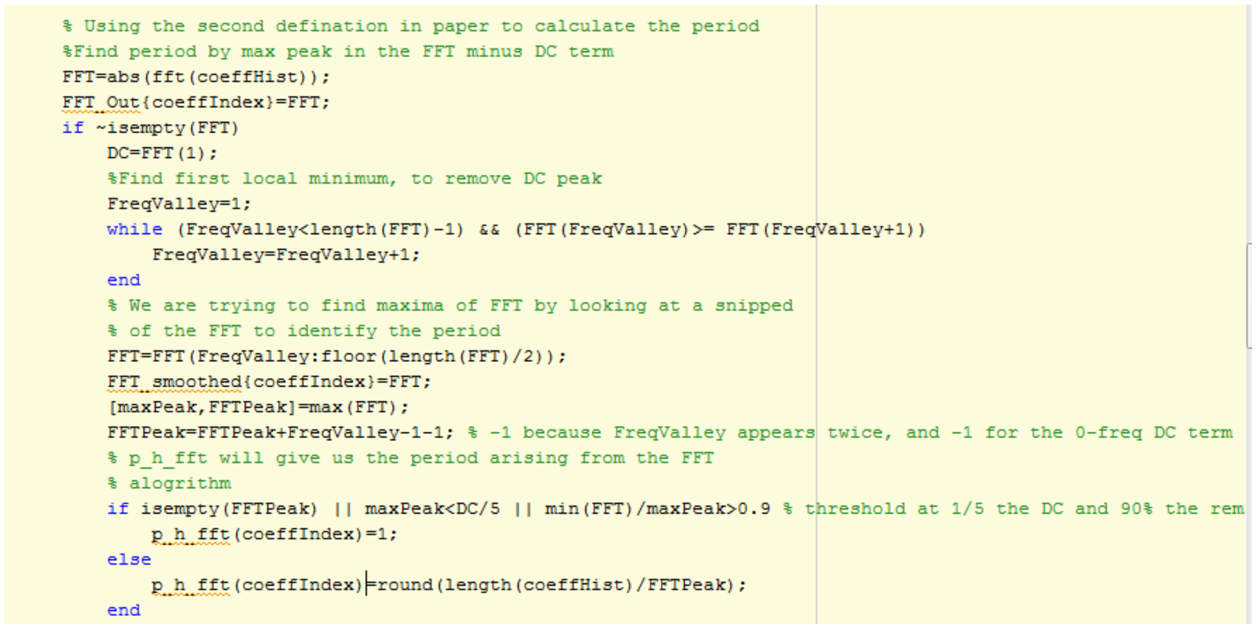
From the figure it is quite intuitive that the H(p) when p is equal to period and hence we can confirm this postulate.

The portion of code that computes this period is given as-



```
%% Calculating the period
        % Using the first defination in paper to calculate the period
        if numel(coeffHist>0)
            %Finding out maximum s_0 coresponding to highest peak in the
            %histogram
            [MaxHVal,s_0]=max(coeffHist);
            s_0_Out(coeffIndex)=s_0;
            dims(coeffIndex)=length(coeffHist);
            H=zeros(floor(length(coeffHist)/4),1);
            % varying the index form 1 to s0/20 to find out the period
            for coeffInd=1:(length(coeffHist)-1)
                vals=[coeffHist(s_0:coeffInd:end) coeffHist(s_0-coeffInd:-coeffInd:1)];
                H(coeffInd)=mean(vals);
            end
            H_Out{coeffIndex}=H;
            % The period will be the coeffInd when H(coeffInd will be
            % maximum)
            [~,p_h_avg(coeffIndex)]=max(H);
```

we can use the fast Fourier transform to find the peak of the spectrum of the histogram with the direct current component removed. This gives another estimate pFFT of the period p.
The portion of code that computes this period is given as-



```
% Using the second defination in paper to calculate the period
%Find period by max peak in the FFT minus DC term
FFT=abs(fft(coeffHist));
FFT_Out{coeffIndex}=FFT;
if ~isempty(FFT)
    DC=FFT(1);
    %Find first local minimum, to remove DC peak
    FreqValley=1;
    while (FreqValley<length(FFT)-1) && (FFT(FreqValley)>= FFT(FreqValley+1))
        FreqValley=FreqValley+1;
    end
    % We are trying to find maxima of FFT by looking at a snipped
    % of the FFT to identify the period
    FFT=FFT(FreqValley:floor(length(FFT)/2));
    FFT_smoothed{coeffIndex}=FFT;
    [maxPeak,FFTPeak]=max(FFT);
    FFTPeak=FFTPeak+FreqValley-1-1; % -1 because FreqValley appears twice, and -1 for the 0-freq DC term
    % p_h_fft will give us the period arising from the FFT
    % alogrithm
    if isempty(FFTPeak) || maxPeak<DC/5 || min(FFT)/maxPeak>0.9 % threshold at 1/5 the DC and 90% the rem
        p_h_fft(coeffIndex)=1;
    else
        p_h_fft(coeffIndex)=round(length(coeffHist)/FFTPeak);
    end
```

Bayesian approach to detecting tampered blocks applicable for both papers

From the above analysis, we see that tampered blocks and unchanged blocks have different bias in terms of contributing to the bins of a histogram h: an unchanged one favors the high peaks of h,
while a tampered one tends to contribute randomly to the bins of h.Our inference is based on this key observation.
Suppose a period starts from the s0-bin and ends at the (s0+p-1)th bin, then the possibility of an unchanged block which contributes to that period occurring in the (s0 + i)-bin can be estimated as

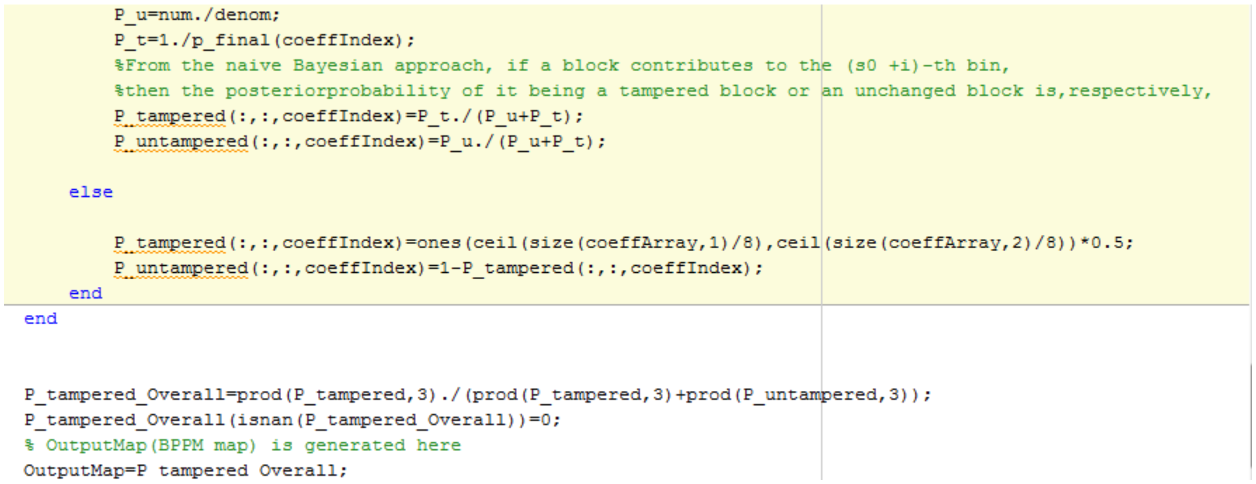
$$p(x|\mathcal{H}_1) = \sum_{L(x)\leq u<R(x)} h_0(u).$$

Here, h(k) denotes the value of the k-th bin of the DCT coefficient histogram h. On the other hand, the possibility of a tampered block which contributes to that period appearing in the bin (s0 + i) can be estimated as

$$P_t(s_0+i) = 1/p$$

due to its randomness of contribution. From the naive Bayesian approach, if a block contributes to the (s0 +i)-th bin, then the posterior probability of it being a tampered block or an unchanged block is,
P(tampered|s0 + i) = Pt/(Pt + Pu)
P(unchanged|s0 + i) = Pu/(Pt + Pu)

The portion of code that computes this period is given as-
For paper 1



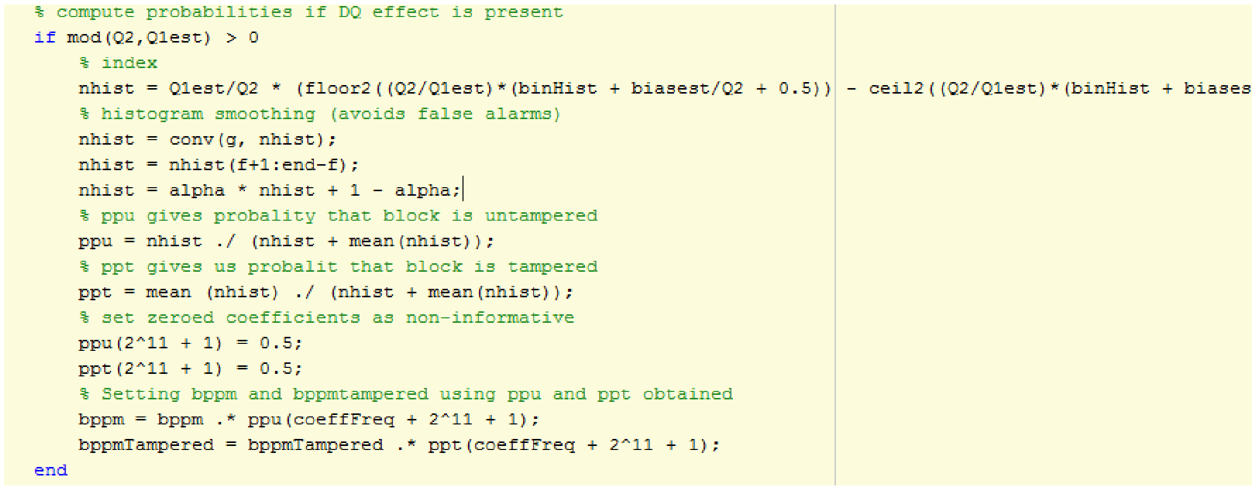
```
        P_u=num./denom;
        P_t=1./p_final(coeffIndex);
        %From the naive Bayesian approach, if a block contributes to the (s0 +i)-th bin,
        %then the posteriorprobability of it being a tampered block or an unchanged block is,respectively,
        P_tampered(:,:,coeffIndex)=P_t./(P_u+P_t);
        P_untampered(:,:,coeffIndex)=P_u./(P_u+P_t);

    else

        P_tampered(:,:,coeffIndex)=ones(ceil(size(coeffArray,1)/8),ceil(size(coeffArray,2)/8))*0.5;
        P_untampered(:,:,coeffIndex)=1-P_tampered(:,:,coeffIndex);
    end
end


P_tampered_Overall=prod(P_tampered,3)./(prod(P_tampered,3)+prod(P_untampered,3));
P_tampered_Overall(isnan(P_tampered_Overall))=0;
% OutputMap(BPPM map) is generated here
OutputMap=P tampered Overall;
```

For paper 2



```
% compute probabilities if DQ effect is present
if mod(Q2,Q1est) > 0
    % index
    nhist = Q1est/Q2 * (floor2((Q2/Q1est)*(binHist + biasest/Q2 + 0.5)) - ceil2((Q2/Q1est)*(binHist + biases
    % histogram smoothing (avoids false alarms)
    nhist = conv(g, nhist);
    nhist = nhist(f+1:end-f);
    nhist = alpha * nhist + 1 - alpha;
    % ppu gives probality that block is untampered
    ppu = nhist ./ (nhist + mean(nhist));
    % ppt gives us probalit that block is tampered
    ppt = mean (nhist) ./ (nhist + mean(nhist));
    % set zeroed coefficients as non-informative
    ppu(2^11 + 1) = 0.5;
    ppt(2^11 + 1) = 0.5;
    % Setting bppm and bppmtampered using ppu and ppt obtained
    bppm = bppm .* ppu(coeffFreq + 2^11 + 1);
    bppmTampered = bppmTampered .* ppt(coeffFreq + 2^11 + 1);
end
```

Feature extraction for paper 1

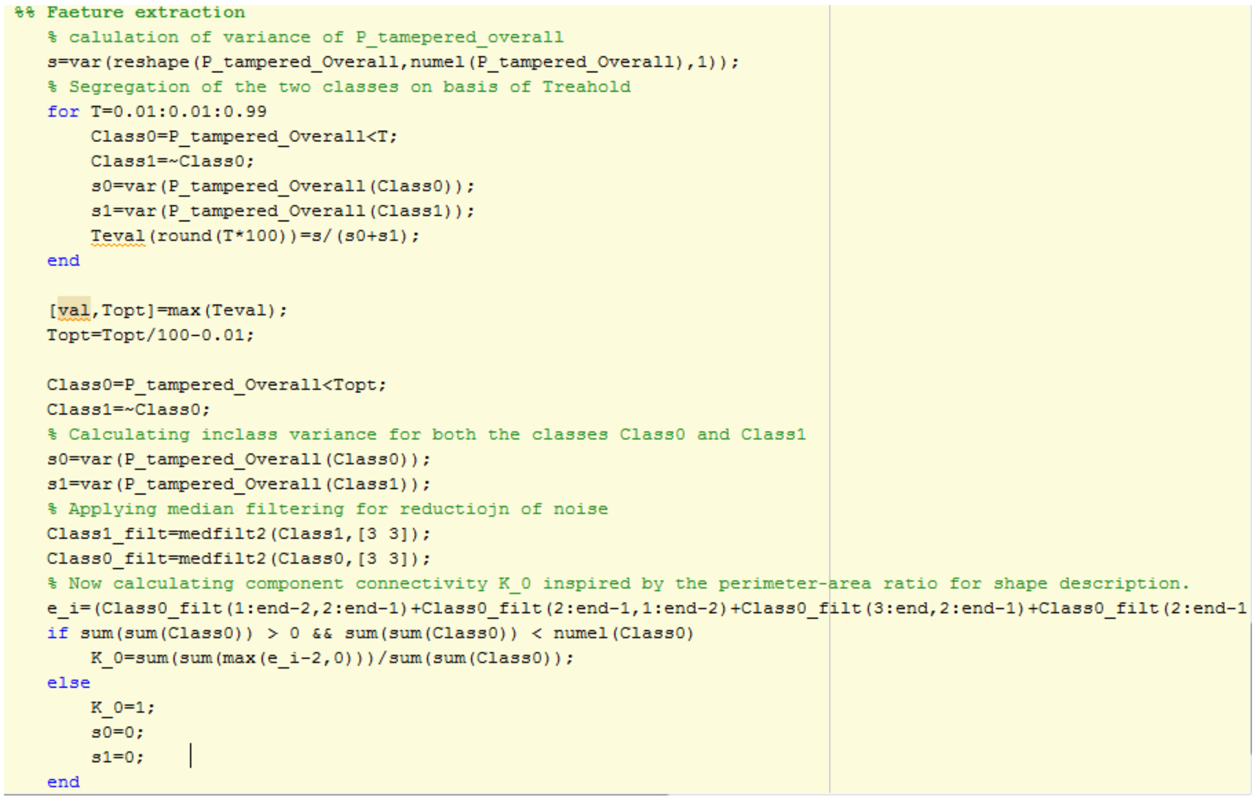
If the image is tampered, we expect that tampered blocks cluster, i.e., the BPPM should be segmented into a small number of regions, where each region has a high probability of being either unchanged or tampered. While any image segmentation algorithm can be applied to the BPPM, to save computation time, we simply threshold the BPPM by choosing a threshold:

$T_{opt} = \arg\max_{T}(\sigma/(\sigma_0 + \sigma_1)),$

where given a T, the pixels of the BPPM are classified into classes C0 and C1, respectively.
in each class, respectively, and is the squared difference between the mean probabilities of the classes.
With the optimal threshold, we expect that those pixels in class C0 (i.e., those having probabilities below Topt) correspond to the tampered blocks in the image.

However, this is still insufficient for confident decision because any BPPM can be segmented in the above
manner as long as its variance is nonzero. Based on the segmentation, we can extract four features: Topt
 and the connectivity K0 of C0. We need the connectivity of C0 as a feature because we expect that the tampered blocks cluster if they exist..

First, the BPPM is denoised by using a medium filter. Then, for each pixel i in C0, find the number ei of pixels in class C1 in its four neighborhood. Finally, we compute K0=
$K_0 = \sum_i \max(e_i - 2, 0)/|C_0|$.
This definition is inspired by the perimeter–area ratio for shape description.
With the four-dimensional feature vector , we can proceed to decide whether the image is tampered, by feeding the feature vector into a trained SVM. If the output is positive, then
the DCT blocks that correspond to C0 of the BPPM are decided as the
tampered region of the image .
Corresponding code for feature extraction

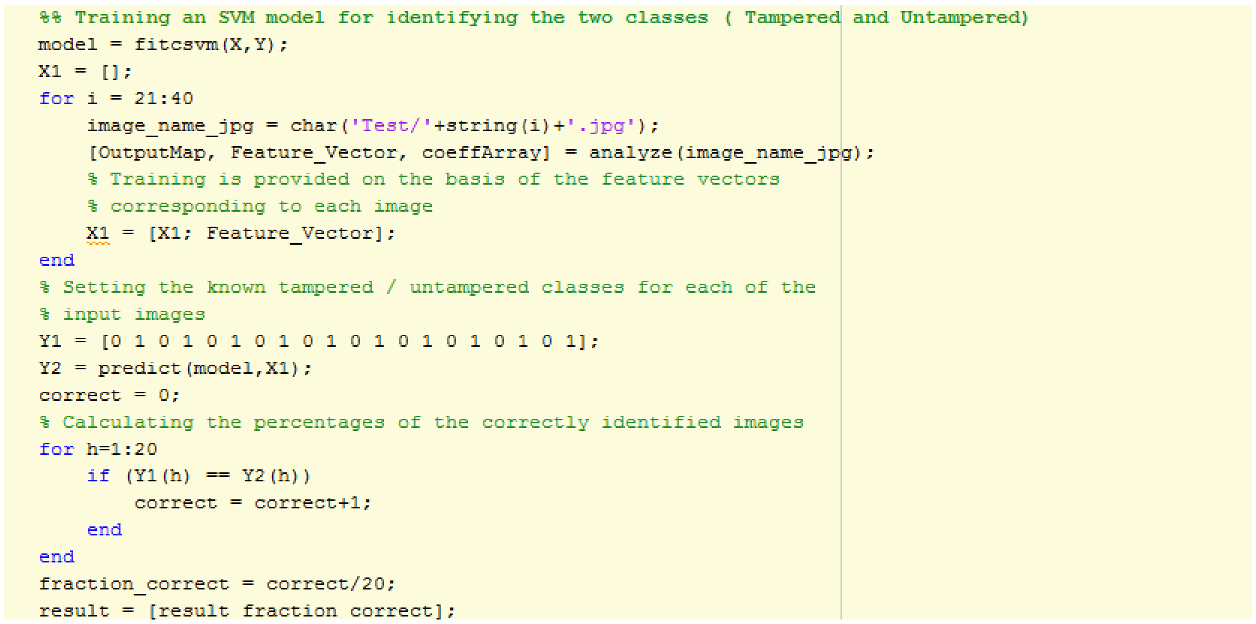


```
%% Faeture extraction
   % calulation of variance of P_tamepered_overall
   s=var(reshape(P_tampered_Overall,numel(P_tampered_Overall),1));
   % Segregation of the two classes on basis of Treahold
   for T=0.01:0.01:0.99
       Class0=P_tampered_Overall<T;
       Class1=~Class0;
       s0=var(P_tampered_Overall(Class0));
       s1=var(P_tampered_Overall(Class1));
       Teval(round(T*100))=s/(s0+s1);
   end

   [val,Topt]=max(Teval);
   Topt=Topt/100-0.01;

   Class0=P_tampered_Overall<Topt;
   Class1=~Class0;
   % Calculating inclass variance for both the classes Class0 and Class1
   s0=var(P_tampered_Overall(Class0));
   s1=var(P_tampered_Overall(Class1));
   % Applying median filtering for reductiojn of noise
   Class1_filt=medfilt2(Class1,[3 3]);
   Class0_filt=medfilt2(Class0,[3 3]);
   % Now calculating component connectivity K_0 inspired by the perimeter-area ratio for shape description.
   e_i=(Class0_filt(1:end-2,2:end-1)+Class0_filt(2:end-1,1:end-2)+Class0_filt(3:end,2:end-1)+Class0_filt(2:end-1
   if sum(sum(Class0)) > 0 && sum(sum(Class0)) < numel(Class0)
       K_0=sum(sum(max(e_i-2,0)))/sum(sum(Class0));
   else
       K_0=1;
       s0=0;
       s1=0;
   end
```

Corresponding code for SVM training



```
%% Training an SVM model for identifying the two classes ( Tampered and Untampered)
model = fitcsvm(X,Y);
X1 = [];
for i = 21:40
    image_name_jpg = char('Test/'+string(i)+'.jpg');
    [OutputMap, Feature_Vector, coeffArray] = analyze(image_name_jpg);
    % Training is provided on the basis of the feature vectors
    % corresponding to each image
    X1 = [X1; Feature_Vector];
end
% Setting the known tampered / untampered classes for each of the
% input images
Y1 = [0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1];
Y2 = predict(model,X1);
correct = 0;
% Calculating the percentages of the correctly identified images
for h=1:20
    if (Y1(h) == Y2(h))
        correct = correct+1;
    end
end
fraction_correct = correct/20;
result = [result fraction_correct];
```

Proposed DCT coefficient analysis for the second paper

In the case of a tampered image, however, a histogram will actually be a mixture of p(x|H1) and p(x|H0). Hence, for large forgeries we expect the histogram of x to be a poor estimate of p(x|H1).
In order to overcome this limitation, we should be able to separate the two conditional probabilities from the observed mixture. By assuming that the histogram h0(x) of the DCT coefficients before the first JPEG compression is available, a better estimate of p(x|H1) could be obtained as

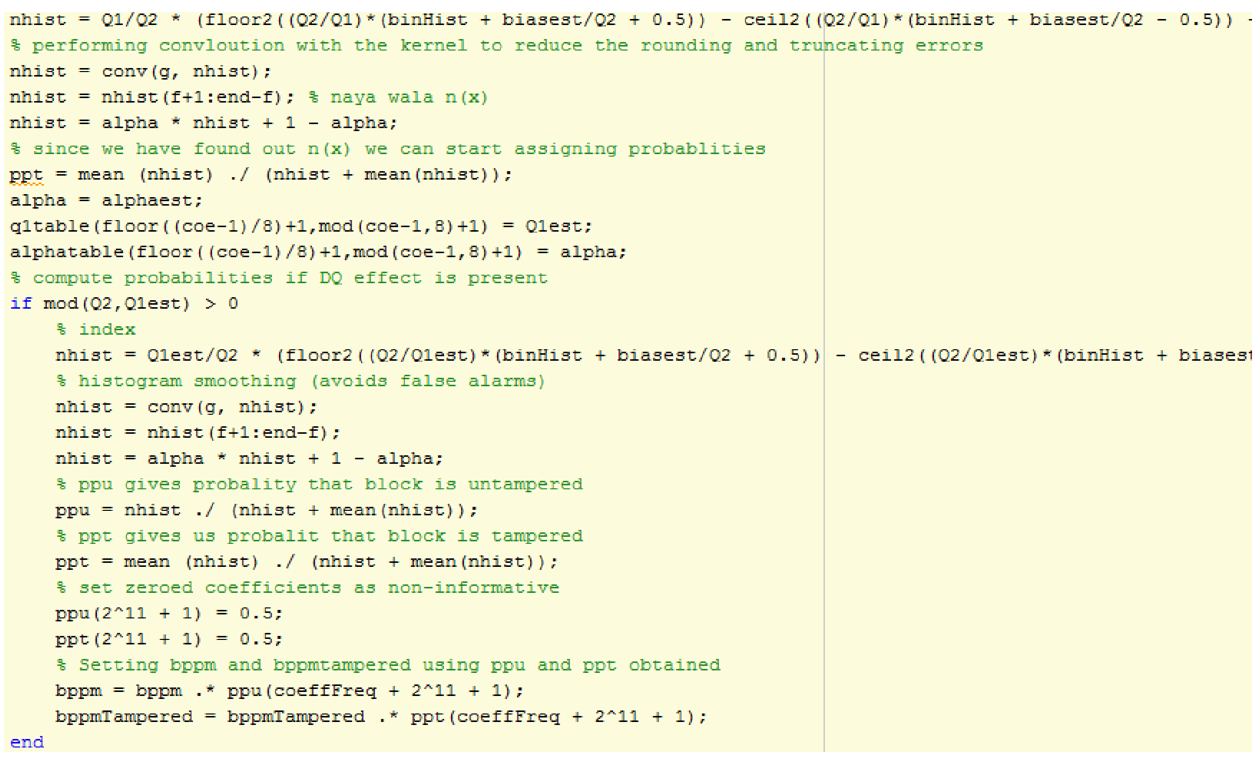
$$p(x|\mathcal{H}_1) = \sum_{L(x)\le u<R(x)} h_0(u).$$

Unfortunately, this equation is difficult to use in practice, since it would require a reliable estimate of both h0(x) and Q1. Hence, it was proposed to introduce the following approximation

$$\frac{1}{n(x)} \sum_{L(x)\le u<R(x)} h_0(u) \approx \frac{1}{Q_2} \sum_{L'(x)\le u<R'(x)} h_0(u) \triangleq \tilde{h}(x)$$

The above approximation holds whenever n(x) > 0 and the histogram of the original DCT coefficient is locally uniform. In practice,it was found that for moderate values of Q2 this is usually true, except for the center bin (x = 0) of the AC coefficients, which have a Laplacian-like distribution.h˜(x) can be viewed as the histogram of the DCT coefficients after a single compression with quantization step Q2. A simple technique for estimating h˜(x) is to consider the DCT coefficients obtained by recompressing with the second quantization matrix a slightly cropped version of the tampered image .

The portion of code that does this is:



```
nhist = Q1/Q2 * (floor2((Q2/Q1)*(binHist + biasest/Q2 + 0.5)) - ceil2((Q2/Q1)*(binHist + biasest/Q2 - 0.5)) -
% performing convloution with the kernel to reduce the rounding and truncating errors
nhist = conv(g, nhist);
nhist = nhist(f+1:end-f); % naya wala n(x)
nhist = alpha * nhist + 1 - alpha;
% since we have found out n(x) we can start assigning probablities
ppt = mean (nhist) ./ (nhist + mean(nhist));
alpha = alphaest;
q1table(floor((coe-1)/8)+1,mod(coe-1,8)+1) = Q1est;
alphatable(floor((coe-1)/8)+1,mod(coe-1,8)+1) = alpha;
% compute probabilities if DQ effect is present
if mod(Q2,Q1est) > 0
    % index
    nhist = Q1est/Q2 * (floor2((Q2/Q1est)*(binHist + biasest/Q2 + 0.5)) - ceil2((Q2/Q1est)*(binHist + biases
    % histogram smoothing (avoids false alarms)
    nhist = conv(g, nhist);
    nhist = nhist(f+1:end-f);
    nhist = alpha * nhist + 1 - alpha;
    % ppu gives probality that block is untampered
    ppu = nhist ./ (nhist + mean(nhist));
    % ppt gives us probalit that block is tampered
    ppt = mean (nhist) ./ (nhist + mean(nhist));
    % set zeroed coefficients as non-informative
    ppu(2^11 + 1) = 0.5;
    ppt(2^11 + 1) = 0.5;
    % Setting bppm and bppmtampered using ppu and ppt obtained
    bppm = bppm .* ppu(coeffFreq + 2^11 + 1);
    bppmTampered = bppmTampered .* ppt(coeffFreq + 2^11 + 1);
end
```

Determination of Q1

It is interesting to note that for determination of n(x) we need Q1 which we can estimate by the following proposal made in paper.

$$p(x; Q_1, \alpha) = \alpha \cdot n(x; Q_1) \cdot \tilde{h}(x) + (1 - \alpha) \cdot \tilde{h}(x)$$

where $\boldsymbol{\alpha}$ is the mixture parameter and we have highlighted the dependence of both p(x) and n(x) from Q1. Based on the above model,
The actual value of Q1 can be estimated as

$$\hat{Q}_1 = \arg\min_{Q_1} \sum_{x \neq 0} [h(x) - p(x; Q_1, \alpha_{opt})]^2$$

This is just the least square and the relation between $\boldsymbol{\alpha}$ and Q1 can be made as-

$$\rightarrow \hat{\theta}_1 = \arg \min_{\theta_1} \sum_{x \neq 0} \left[h(x) - p(x; \theta_1, \alpha_{opt}) \right]^2$$

$$\frac{\partial \hat{\theta}_1}{\partial \alpha} = 0 \quad [\text{for best } \alpha]$$

$$\Rightarrow \quad 2 \sum_{x \neq 0} \left[h(x) - p(x; \theta_1, \alpha) \right] \left[- \frac{\partial p(x; \theta_1, \alpha)}{\partial \alpha} \right] = 0$$

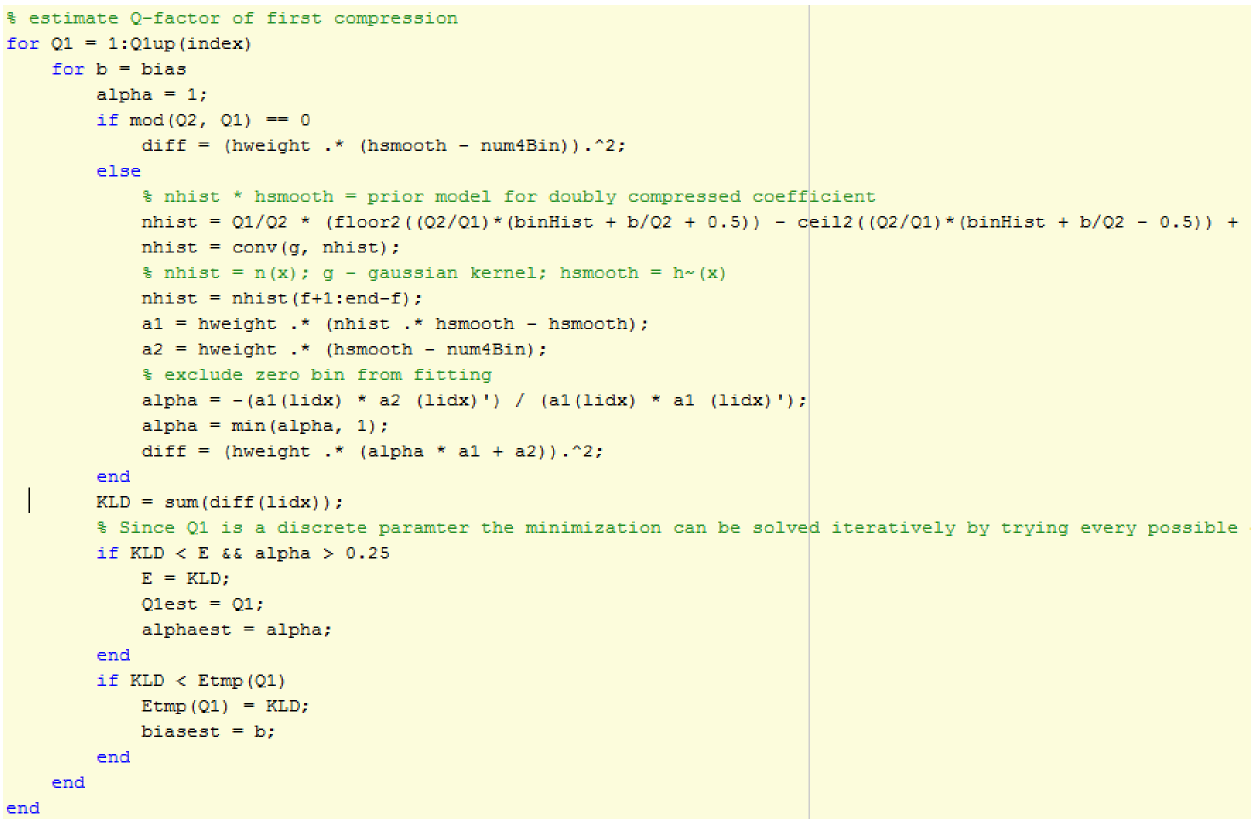
$$p(x; \theta_1, \alpha) = \alpha \, n(x; \theta_1) \, \bar{h}(x) + (1 - \alpha) \, \bar{h}(x)$$

$$\frac{\partial p(x; \theta_1, \alpha)}{\partial \alpha} = n(x; \theta_1) \, \bar{h}(x) - \bar{h}(x)$$

$$\Rightarrow \sum_{x \neq 0} \left[h(x) - \alpha \, n(x; \theta_1) \bar{h}(x) + (\alpha - 1) \, \bar{h}(x) \right] \bar{h}(x) \left[n(x; \theta_1) - 1 \right] = 0$$

$$\Rightarrow - \frac{\sum_{x \neq 0} \left[h(x) - \bar{h}(x) \right] \bar{h}(x) \left[n(x; \theta_1) - 1 \right]}{\sum_{x \neq 0} \bar{h}(x)^2 \left[n(x; \theta) - 1 \right]^2} = \alpha_{opt}$$

The portion of code that does this is-

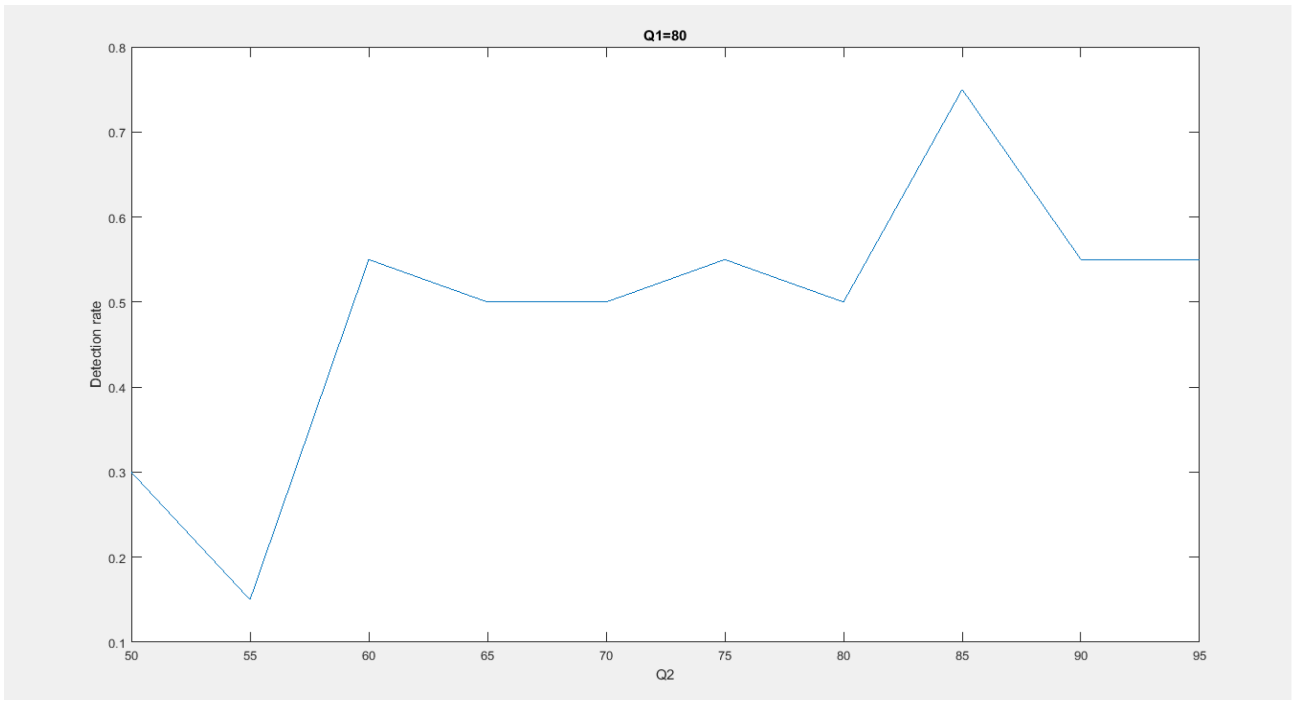


```
% estimate Q-factor of first compression
for Q1 = 1:Q1up(index)
    for b = bias
        alpha = 1;
        if mod(Q2, Q1) == 0
            diff = (hweight .* (hsmooth - num4Bin)).^2;
        else
            % nhist * hsmooth = prior model for doubly compressed coefficient
            nhist = Q1/Q2 * (floor2((Q2/Q1)*(binHist + b/Q2 + 0.5)) - ceil2((Q2/Q1)*(binHist + b/Q2 - 0.5)) +
            nhist = conv(g, nhist);
            % nhist = n(x); g - gaussian kernel; hsmooth = h~(x)
            nhist = nhist(f+1:end-f);
            a1 = hweight .* (nhist .* hsmooth - hsmooth);
            a2 = hweight .* (hsmooth - num4Bin);
            % exclude zero bin from fitting
            alpha = -(a1(lidx) * a2 (lidx)') / (a1(lidx) * a1 (lidx)');
            alpha = min(alpha, 1);
            diff = (hweight .* (alpha * a1 + a2)).^2;
        end
        KLD = sum(diff(lidx));
        % Since Q1 is a discrete paramter the minimization can be solved iteratively by trying every possible
        if KLD < E && alpha > 0.25
            E = KLD;
            Q1est = Q1;
            alphaest = alpha;
        end
        if KLD < Etmp(Q1)
            Etmp(Q1) = KLD;
            biasest = b;
        end
    end
end
```

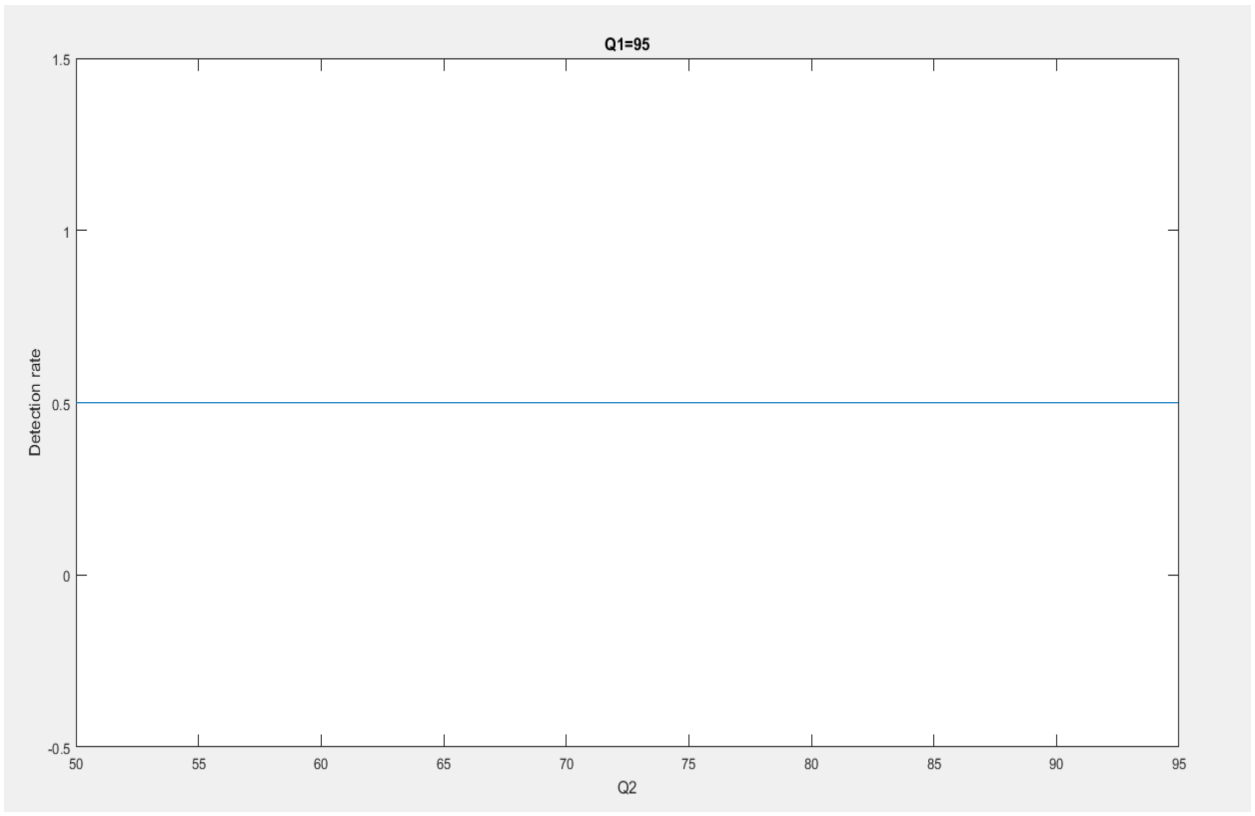
Results and discussion

1) For the paper 1

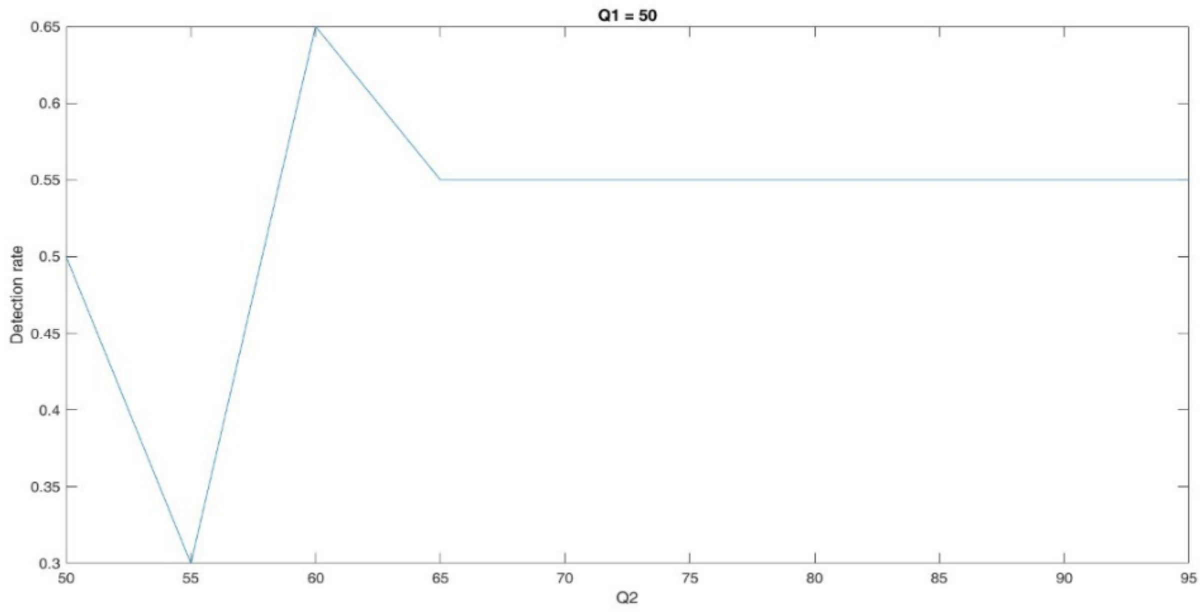
We have written the code for training an SVM model with images corresponding to QF1 From 50 to 95 at a step of 5 and QF2 From 50 to 95 at a step of 5 for 20 iterations generating a tempered and untempered image at each iteration thus generating a total of 4000 images for training SVM and then used this Model for detecting 20 further images as forged or unforged but this requires a lot of computation power so instead we wrote code demo1.m which trains SVM with a fixed QF1 and varying QF2 with a step size of 5 from 50 to 95 and took i as 1 to train SVM(Giving 1 tampered and 1 untampered image into SVM) . Then we tried detecting the 20 images as forged or unforged and plotted the percentage of correct detections with varying Q2 keeping Q1 fixed and these were the results obtained:



For Q1=80 and varying Q2

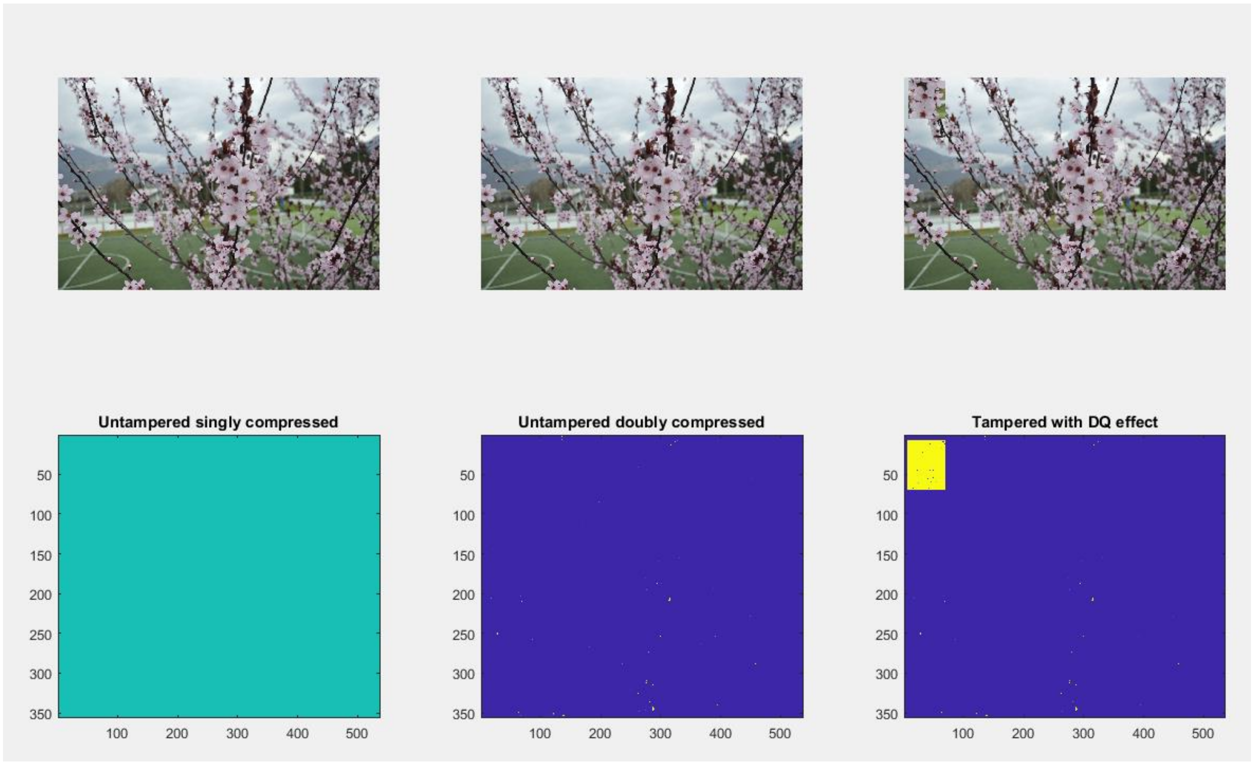


For Q1 =95 and varying Q2



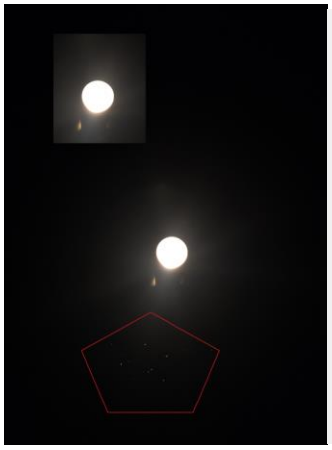
For Q1=50 and varying Q2

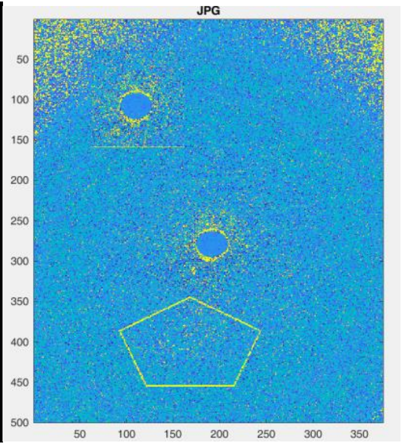
Even to perform these tasks an average of 30 mins for implementation for a i5 processor.



Our results almost match the output given by the paper and hence we have replicated their results.

Drawbacks
The result of untampered singly compressed is given as a 0.5 probability map and does not clearly segregate this untampered image.
The result of the image given here was not detected properly and forgery of this type might go unnoticed.





If a forgery is introduced at 8*8 multiple position then it might be detected but as discussed with bhaiya that has a probability of just(1/64).
Also the forgery might be introduced from a doubly compressed image and then detection will be very difficult.

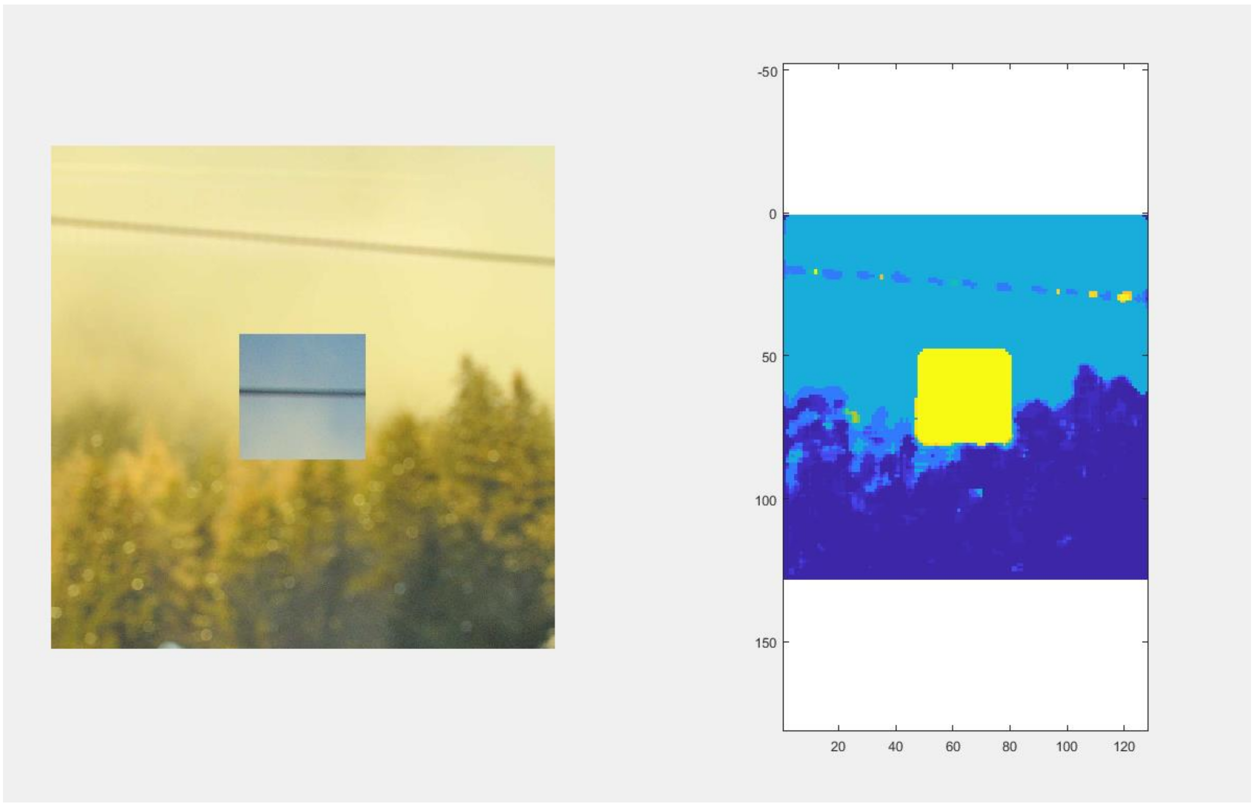
2) For the paper 2

We have written the code for calculating AUC of ROC curve with images corresponding to QF1 From 50 to 100 at a step of 10 and QF2 From 50 to 100 at a step of 10 with varying the threshold form 0 to 1 at a step of 0.00001 for 20 iterations of tampered images. All images have been taken of size 1024*1024 and the central portion of size 256 × 256 is then replaced with the corresponding area from the original TIFF image.finally, the overall “manipulated” image is JPEG compressed (again with Matlab) with another given quality factor QF2. In this way, the image will result doubly compressed everywhere, except in the central
region where it is supposed to be forged. Both the considered algorithms provide as output, for each analyzed image, a probability map that represents the probability of each 8 × 8 block to be forged (i.e. for each 1024 × 1024i image a 128 × 128 probability map is given).
For a particular threshold we determine pfa(false alarm rate) and pd (missed detection probability) which form the basis of the ROC curve and the area under this curve is calculated. Then the mean auc is calculated for each of the 36 combinations .
But this requires a lot of computation power so instead we also wrote a code as demo1.m which takes iterations for a particular Q1 and Q2 for 1 iteration of image and then AUC is calculated and ROC is plotted. We do this for 5 image and take their mean and we were able to match the results as dictated in the paper.

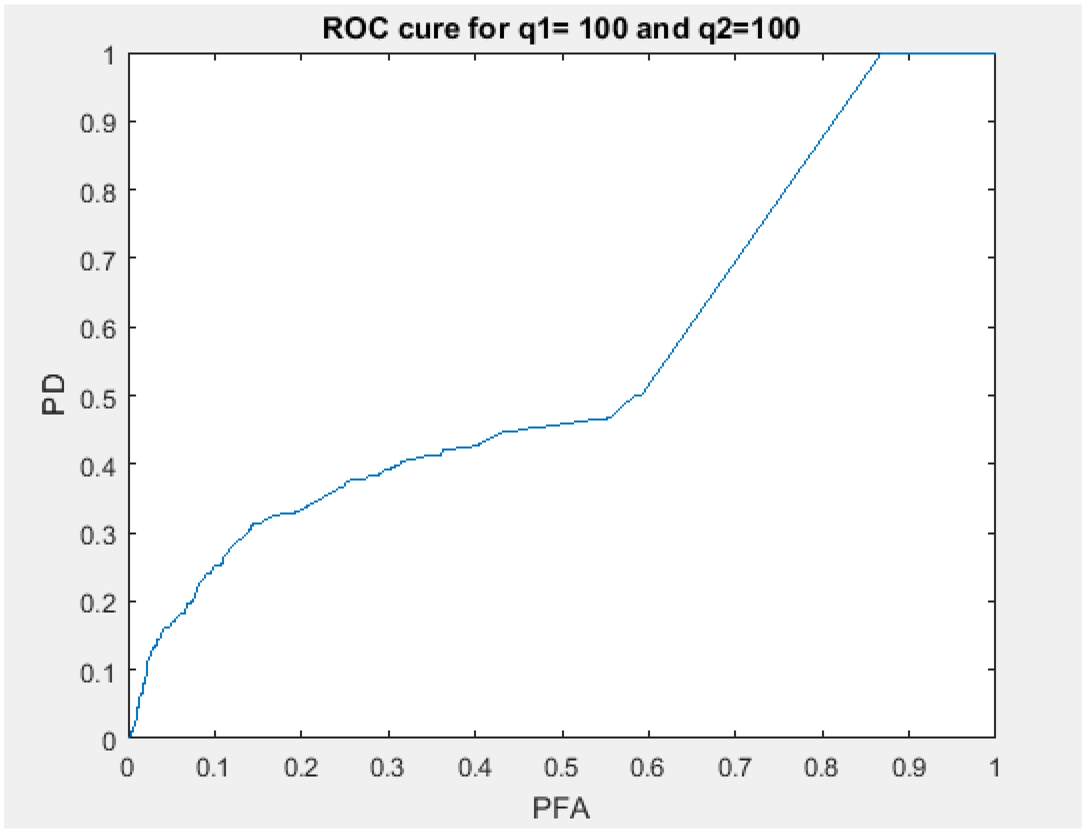
			AUC characteristic for different Q1 and Q2 for 5 image set						
	Q1=50,Q	Q1=50,Q	Q1=50,Q	Q1=80,Q	Q1=80,Q	Q1=80,Q	Q1=100,	Q1=100,	Q1=100,

	2=50	2=80	2=100	2=50	2=80	2=100	Q2=50	Q2=80	Q2=100
Image 1	0.6173	0.9974	0.9994	0.5508	0.6851	0.9986	0.6715	0.6963	0.5004
Image 2	0.63	0.9965	0.991	0.5671	0.65	0.9989	0.6814	0.7104	0.4981
Image 3	0.61	0.9953	0.996	0.531	0.6712	0.9981	0.6914	0.6894	0.501
Image 4	0.58	0.998	0.994	0.546	0.691	0.9991	0.6413	0.6931	0.5007
Image 5	0.6134	0.991	0.991	0.561	0.63	0.9986	0.6614	0.7034	0.5004
Mean Value obtained	0.61014	0.99564	0.99428	0.55118	0.66546	0.99866	0.6694	0.69852	0.50012

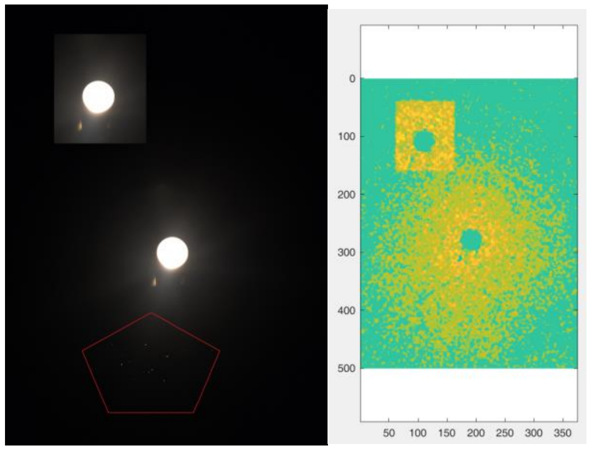
The tampering map example is given in the following which the highlighted yellow portion indicates forgery.



ROC curve example



Advantages over Algorithm1:
It even works well when QF1 > QF2 which is not in the case of algorithm 1.
It does require to train a SVM but rather makes decision on the basis of the optimum threshold value which is kept near 0.52 which generates the best result.
Drawback :
Also the forgery might be introduced from a doubly compressed image and then detection will be very difficult.
The result of the image given here was not detected properly and forgery of this type might go unnoticed.



Learnings:

1) Basis of Forgery detection.

2) SVM method being used as a classifier.
3) Learnt about ROC and significance of AUC curve.
4) DQ effect and its significance for forgery.
5) Bayesian classifiers.
6) A prominent method for feature extraction
7) Use of gaussian kernels for removing the R/T errors.
8) Proposed DCT coefficient analysis