Video transmission over the internet is common nowadays because of the availability of large bandwidth. Video files have large amount of redundancy, so huge amount of payload can be embedded. The distortion is unnoticeable here.

In a video file, the video and the audio frames are stored separately and synchronization is used during decoding. There are three types of video frames: I, P, and B. An I-frame, or intra frame, is a self-contained frame that can be independently decoded without any reference to other images. The first image in a video sequence is always an I-frame. A P-frame, which stands for predictive inter frame, makes references to parts of earlier I- and/or P-frame(s) to code the frame. It contains motion-compensated difference information relative to previous I- or P-frames. A B-frame, or bi-predictive inter frame, is a frame that makes references to preceding and succeeding I- or P-frames. Therefore, P-frames and B-frames are only concerned about capturing the motion. A number of I-, P- and B-frames together is called as Group of Pictures or GOP. It starts with an I-frame. Its length (number of frames between two I-frames, excluding the second I frame) varies depending on the codec used. A typical GOP structure is IBBPBBPBBBPBB or IBBBPBBBBB. One GOP of length 9 is shown in Fig. 6.1. The I-frame is used to predict the first P-frame and these two frames are also used to predict the first and the second B-frame. The second P-frame is predicted using the first P-frame and they join to predict the third and fourth B-frames.
This chapter describes three video steganography techniques considering compressed and uncompressed video as cover. The common compressed MP4 video and uncompressed AVI video are used as cover video. The secret information considered are image and audio since video files have large space for information hiding. The performance of the techniques is analyzed by considering the metrics PSNR, SSIM, CQM, VQM, SNR and SPCC. The effect of the common attacks such as RST, compression, insertion and deletion of frames, change of frame rate, frame interchanging etc. is discussed.

### 6.1 Hiding Images in Video

In this Video Steganography (VS) technique, the secret images are hidden in a video using IWT and it is named as VSIWT-IIV (IIV is the abbreviation for Image in Video). MP4 video file is taken as the cover and multiple secret images are hidden in the cover video. The stego video should be in uncompressed form, which is the main drawback of this technique. The results are discussed at the end of this section.

The frames are extracted from the video and used for steganography. Since the frames are still images, the image steganography technique can be used. VideoReader() and read() functions of MATLAB are used for reading a video file and extracting a particular frame respectively. Two grayscale secret images are hidden in one frame using ISIWT-GIC technique explained in section 4.2. A random frame is read and it is converted to YCbCr color space. Two secret images are hidden: one in Cb and one in Cr component. The embedding and extracting algorithms are given in Figs. 6.2 and 6.3 respectively. The block diagram of the embedding procedure is shown in Fig. 6.2a.

<table>
<thead>
<tr>
<th>Algorithm: Embed-(VSIWT-IIV). Embeds the secret images in the video.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>• Cover video C.mp4</td>
</tr>
<tr>
<td>• Secret images S1.jpg and S2.jpg</td>
</tr>
<tr>
<td><strong>Output:</strong> Stego audio G.avi.</td>
</tr>
<tr>
<td><strong>Method:</strong></td>
</tr>
<tr>
<td>1. ( \text{vid} \leftarrow \text{VideoReader('C.mp4')} )</td>
</tr>
</tbody>
</table>
2. vidFrames ← read(vid)
3. S1 ← imread('football_gray.jpg')
4. S2 ← imread('earth1.jpg') // Read cover video and secret images.
5. C ← read(vid, 1000)
6. CyCbCr ← rgb2ycbcr(C) // Extract one random frame say 1000th frame and Represent C // in YCbCr color space. Two secret images are hidden - one inCb and one in Cr \ // components.
7. LS ← liftwave('haar', 'Int2Int')
8. [CbLL, CbHL, CbLH, CbHH] ← lwt2(double(Cb), LS)
9. [CrLL, CrHL, CrLH, CrHH] ← lwt2(double(Cr), LS) // Obtain IWT of Cb and Cr // components to get CbLL, CbHL, CbLH and CbHH, CrLL, CrHL, CrLH and CrHH
10. [sLL1, sHL1, sLH1, sHH1] ← lwt2(double(S1), LS)
11. [sLL2, sHL2, sLH2, sHH2] ← lwt2(double(S2), LS) // Obtain IWT of the secret images.
12. for i = 1 to size(sLL1) do
13. CbHH(i) ← sLL1(i)
14. endfor // Hide the low frequency sub-band of the secret image S1 by replacing the // coefficients in CbHH band. Also, hide the no. of coefficients hidden.
15. for i = 1 to size(sLL2)
16. CrHH(i) ← sLL2(i)
17. endfor
18. G1 ← ilwt2(CbLL, CbHL, CbLH, CbHH, LS)
19. G2 ← ilwt2(CrLL, CrHL, CrLH, CrHH, LS) // Obtain the inverse IWT to get Cb and // Cr components
20. G ← [CyG1G2] // Integrate the Cy, G1, and G2 components into single image frame
21. C ← ycbcr2rgb(G) // Convert back to RGB color space
22. vidFrames(:, :, :, 1000) ← C // Add the frame to the video in the correct location
23. nFrames ← vid.NumberOfFrames
24. mov(1:nFrames) ← struct('cdata', zeros(vidHeight, vidWidth, 3, 'uint8'), 'colormap', [])
25. mov(1000).cdata ← vidFrames(:, :, :, 1000)
26. writeObj ← VideoWriter('G')
27. writeObj.FrameRate=vid.FrameRate
28. open(writeObj)
29. writeVideo(writeObj, mov)
30. close(writeObj) // Store the video in the AVI format.
31. return stego video G.

Fig. 6.2. VSIWT-IIV embedding algorithm (continued in the next page)
**Algorithm: Extract-(VSIWT-IIV).** Extracts the secret images from the stego video.

**Input:** Stego video G.avi

**Output:** Secret images S1.jpg and S2.jpg.

**Method:**
1. `vid ← VideoReader('G.avi')`
2. `vidFrames ← read(vid)` \// Read stego video.
3. `Cg ← vidFrames(:, :, :, 1000)`
4. `Cy1Cb1Cr1 ← rgb2ycbcr(Cg)` \// Extract the frame in which secret images are hidden and convert to YCbCr color space.
5. `LS ← liftwave ('haar', 'Int2Int')`
6. `[gLL1,gHL1,gLH1,gHH1] ← lwt2(double(Cb1), LS)`
7. `[gLL2,gHL2,gLH2,gHH2] ← lwt2(double(Cr1), LS)` \// Obtain IWT of Cb and Cr // components
8. `for i=1 to secretsize/4 do`
9. `newSLL1(i) ← gHH1(i)`
10. `newSLL2(i) ← gHH2(i)`
11. `endfor` \// Extract the low frequency sub-bands of the secret images hidden in the high // frequency sub-bands of Cb and Cr components.
12. `S1 ← ilwt2(newSLL1, 0, 0, 0, LS)`
13. `S2 ← ilwt2(newSLL2, 0, 0, 0, LS)` \// Obtain the inverse IWT by considering zeroes for // high frequency components to get secret images.
14. `return secret image S1.jpg and S2.jpg.`

Fig. 6.3. VSIWT-IIV extracting algorithm

### 6.1.1 Results and discussion

This video steganography technique is tested with an MP4 video clip, downloaded from https://www.youtube.com/watch?v=3yFiqdCjNMk. A frame of this video clip
and two secret images are shown in Fig. 6.4. The frame size is 640 x 360 and the secret images are of size 128 x 128. The results are analyzed, taking 1, 2, and 3 frames as cover images. The metrics PSNR, SSIM, and CQM are computed for each of the stego frames and the average value is taken as the performance metric. The stego frame and the extracted secret images are shown in Fig. 6.5. Table 6.1 shows the performance metrics for the stego video and the extracted secret images. For any number of frames used as cover frames there is no perceptible difference between the cover video and the stego video.

![Fig. 6.4. Inputs (VSIWT-IIV): (a) One of the frames from the video (b) Secret image football (c) Secret image earth (VSIWT-IIV)](image)

![Fig. 6.5. Stego frame and Extracted images (VSIWT-IIV): (a) stego frame (b) extracted image football (c) extracted image earth](image)

Table 6.1. Performance metrics for video steganography (VSIWT-IIV)
<table>
<thead>
<tr>
<th>Cover video</th>
<th>Secret images (grayscale)</th>
<th>No. of frames used</th>
<th>Stego PSNR in dB</th>
<th>Stego SSIM</th>
<th>Stego CQM in dB</th>
<th>Extracted PSNR in dB</th>
<th>Extracted SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>video.mp4</td>
<td>Two images with size 128x128</td>
<td>1</td>
<td>36.4</td>
<td>0.9501</td>
<td>38.2</td>
<td>S1:36.2</td>
<td>S1:0.9949</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S2:36.1</td>
<td>S2:0.9053</td>
</tr>
<tr>
<td></td>
<td>Four images with size 128x128</td>
<td>2</td>
<td>35.5</td>
<td>0.9325</td>
<td>37.5</td>
<td>S1:36.2</td>
<td>S1:0.9949</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S2:36.1</td>
<td>S2:0.9053</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S3:35.9</td>
<td>S3:0.9894</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S4:32.4</td>
<td>S4:0.8812</td>
</tr>
<tr>
<td></td>
<td>Six images with size 128x128</td>
<td>3</td>
<td>35</td>
<td>0.9132</td>
<td>36.1</td>
<td>S1:36.2</td>
<td>S1:0.9949</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S2:36.1</td>
<td>S2:0.9053</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S3:35.9</td>
<td>S3:0.9894</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S4:32.4</td>
<td>S4:0.8812</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S5:36.1</td>
<td>S5:0.9907</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S6:34.4</td>
<td>S6:0.9355</td>
</tr>
</tbody>
</table>

### 6.1.2 Performance Against Attacks

For video steganography, common attacks are RST (Rotation, Scaling, and Translation), compression, change of frame rate, interchange of frames, addition or deletion of frames etc. VSIWT-IIIV withstands RST, but fails all other attacks. Therefore, it is modified and is discussed in the next section. With a RST attack, the secret images are extracted and the performance metrics are shown in Table 6.2. The extracted secret images are shown in Fig. 6.6. When the stego frame is rotated by three degrees, the pixels are translated from their original position and the frame size increases. It is scaled to the original size. With these deformations to the stego frame, the secret images can be extracted.
Table 6.2. Performance against common attacks (VSIWT-IIV)

<table>
<thead>
<tr>
<th>Cover video</th>
<th>Secret images (grayscale)</th>
<th>No. of frames used</th>
<th>Stego</th>
<th>Extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infile1.avi</td>
<td>Two images with size 128x128</td>
<td>1</td>
<td>35</td>
<td>0.950</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6.6. Extracted secret images with attack (VSIWT-IIV)

6.2 Hiding Audio and Images in Video

A video file contains both video and audio frames. The Video frames are still images and therefore, can be used for image steganography. When the audio frames are extracted from the video it is like an audio file and it can be used for audio steganography. In this section, two techniques are explained considering AVI and MP4 videos as the cover objects. The performance of the techniques is measured using the metrics PSNR, SSIM, VQM, SNR, and SPCC. The response against the common attacks is also analyzed.

6.2.1 Video Steganography in Uncompressed AVI Files

This is a video steganography technique in uncompressed video streams, which uses uncompressed AVI video file as the cover and hence the technique is named as VSAVI. The secret audio and images are hidden in the audio and the video frames of the video, respectively. The video files containing audio are split to get the video and the audio frames. When a small video clip is read, thousands of video and audio frames are available. All these frames can be used for steganography, but then the stego video file cannot resist compression, even though the other attacks like RST, addition and deletion of frames can be handled efficiently. When
RST attack is applied, the secret data is not lost since it is applied to the entire frame. To handle frame deletion or addition, random frames are selected as per the random number generator and the secret information is hidden in duplicates. Pseudo Random Number Generator (PRNG) principle is used to select the frames. Only the seed has to be exchanged between the sender and the receiver. The first two prime factors of the seed are found out and the sum of these two prime factors gives the first frame number where the secret data is hidden. Then, for the next two continuous frames, the same secret data is hidden. This procedure is repeated three times in the frames at an offset of sum of the prime factors. For example, if 10 is the seed then 2 and 5 are its first two prime factors. Their sum is 7. The secret data is hidden in the continuous frames 7, 8, and 9, followed by 16, 17, and 18 (because 9+7=16) and finally in the frames 23, 24, and 25 (because 16+7=23). At the receiver, the data is extracted from all these frames and checked for majority similarity. There is no chance of losing all such frames. So any addition or deletion of frames can be handled. The audio frame size is small and the first frame always contains a silent period. Therefore, except the first frame, all other frames are combined together to hide the secret audio. The secret audio is hidden in duplicate to resist the frame loss.

The functions VideoFileReader() and step() defined in MATLAB are used for extracting image and audio frames. To hide the secret images, first, the video frames are selected using PRNG as explained above and then the two secret images are hidden in duplicate using the VSIWT-IIV technique presented in section 6.1. To hide the secret audio, first, IWT is applied on both cover audio frames and secret audio and then, the approximate coefficients of the secret audio are embedded in the detailed coefficients of the cover audio. The embedding and extracting algorithms are as given in Figs. 6.7 and 6.8 respectively. The block diagram is shown in Fig. 6.7a.

<table>
<thead>
<tr>
<th>Algorithm: Embed-(VSAVI). Embeds the secret image and audio in the cover video.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>• Cover video <em>C.avi</em></td>
</tr>
<tr>
<td>• Secret images <em>S1.jpg</em> and <em>S2.jpg</em>.</td>
</tr>
<tr>
<td>• Secret audio <em>S3.wav</em>.</td>
</tr>
<tr>
<td><strong>Output:</strong> Stego video, <em>G.avi</em>.</td>
</tr>
</tbody>
</table>
Method:

1. VideoFreader \( \leftarrow \text{vision.VideoFileReader}('C.avi') \)
2. vid \( \leftarrow \text{VideoReader}('C.avi') \)
3. \( n \leftarrow \text{vid.NumberOfFrames} \)
4. for \( i = 1 \) to \( n \) do
   
   5. [I, Audio] \( \leftarrow \text{step(VideoFreader)} \)
   
   6. \( a(i) \leftarrow \text{Audio} \)
   
   7. \( \text{im}(i) \leftarrow I \)

8. endfor \( // \) Read cover video and get video and audio frames. Store audio frames in ‘a’ and video frames in ‘im’.

9. Get the video frame numbers as per the PRNG and hide the secret image in duplicates. Convert the video frame to YCbCr color space and hide two secret images after Arnold transformation: one in Cb and one in Cr components. Hiding procedure in the frame is explained in the following steps.

10. \( LS \leftarrow \text{liftwave('haar','Int2Int')} \)

11. [BLL, BHL, BLH, BHH] \( \leftarrow \text{lwt2(double(C1), LS)} \)

12. [GLL, GHL, GLH, GHH] \( \leftarrow \text{lwt2(double(C2), LS)} // C1 is the Cb component of the //selected frame and C2 is the Cr component of the selected frame. Obtain IWT of Cb and //Cr components to get BLL, BHL, BLH, and BHH, GLL, GHL, GLH, and GHH.

13. [sLL1, sHL1, sLH1, sHH1] \( \leftarrow \text{lwt2(double(S1), LS)} \)

Fig. 6.7. VSAVI embedding algorithm (continued in the next page)

14. [sLL2, sHL2, sLH2, sHH2] \( \leftarrow \text{lwt2(double(S2), LS)} // \) Obtain IWT of the Arnold //transformed secret images.

15. for \( i = 1 \) to size(sLL1) do

16. \( \text{BHH}(i) \leftarrow sLL1(i) \)

17. endfor \( // \) Hide the low frequency sub-band of the secret image S1 by replacing the //coefficients in BHH band. Also, hide the number of coefficients hidden.

18. for \( i=1 \) to size(sLL2) do

19. \( \text{GHH}(i) \leftarrow sLL2(i) \)

20. endfor \( // \) Hide the low frequency sub-band of the secret image S2 by replacing the //coefficients in GHH band along with the number of coefficients hidden.

21. \( G1 \leftarrow \text{ilwt2}(\text{BLL, BHL, BLH, BHH}) \)

22. \( G2 \leftarrow \text{ilwt2}(\text{GLL, GHL, GLH, GHH}) // \) Obtain the inverse IWT to get Cb and Cr //components

23. Integrate the Y, Cb, and Cr components into a single image frame and convert back to RGB.

24. Add the frame to the video in the correct location.

25. Except the first frame, all other audio frames are combined.

26. [CAc, CDc] \( \leftarrow \text{lwt(double(frame), LS)} \)

27. [CAS, CDs] \( \leftarrow \text{lwt(double(S3), LS)} // \) Obtain IWT of the combined audio frames and secret //audio S3 to get approximation and detail coefficients.

28. Obtain the binary of CAS and duplicate the bits three times.
29. Hide the number of CA coefficients in the first detailed coefficient and the duplicated secret bits in the third, fourth and fifth bit planes of the other detailed coefficients of the cover.

30. Obtain the inverse IWT to get the stego audio samples and then convert into frames. Add the frames to the audio in the correct location.

31. \texttt{VideoFwriter} \leftarrow \texttt{vision.VideoFileWriter('G.avi')}

32. \texttt{for} i = 1 to n \texttt{do}

33. \texttt{step(VideoFwriter, im\{i\}, a\{i\})}

34. \texttt{endfor} // Store the video and audio frames in the AVI format in the uncompressed form.

---

Fig. 6.7. VSAVI embedding algorithm
Fig. 6.7a. Block diagram – VSAVI (Embedding)
Algorithm: Extract-(VSAVI). Extracts the secret images and the secret audio from the stego video.

Input: Stego video G.avi

Output:
- Secret images $S1.jpg$ and $S2.jpg$.
- Secret audio $S3.wav$.

Method:
1. $\text{VideoFreader} \leftarrow \text{vision.VideoFileReader}(\text{‘G.avi’})$
2. $\text{vid} \leftarrow \text{VideoReader}(\text{‘G.avi’})$
3. $n \leftarrow \text{vid.NumberOfFrames}$
4. for $i=1$ to $n$ do
5. \hspace{1em} $[\text{I,Audio}] \leftarrow \text{step(VideoFreader)}$
6. \hspace{1em} $a1[i] \leftarrow \text{Audio}$
7. \hspace{1em} $im1[i] \leftarrow 1$
8. endfor //Get the frame numbers as per the PRNG and extract the required video and audio frames. The hidden secret images are extracted as follows:
9. Convert the extracted video frames into YCbCr color space.
10. $[gLL1, gHL1, gLH1, gHH1] \leftarrow \text{lwt2(double(Cb), LS)}$
11. $[gLL2, gHL2, gLH2, gHH2] \leftarrow \text{lwt2(double(Cr), LS)}$ // Obtain IWT of Cb and Cr components and get the size of the secret images.
12. for $i=1$ to $\text{secretsize}/4$ do
13. \hspace{1em} new$SLL1(i) \leftarrow gHH1(i)$
14. \hspace{1em} new$SLL2(i) \leftarrow gHH2(i)$
15. endfor // Extract the low frequency sub-bands of the secret images hidden in the high frequency sub-bands of the secret images hidden in the high frequency sub-bands of the Cb and Cr components of all the selected frames.
16. $S1 \leftarrow \text{ilwt2(newSLL1, 0, 0, 0, LS)}$
17. $S2 \leftarrow \text{ilwt2(newSLL2, 0, 0, 0, LS)}$ // Obtain the inverse IWT by considering zeroes for high frequency components to get secret images.
18. Extract all the duplicates of the secret images similarly. Then the corresponding secret images are compared for majority similarity and that is selected as the secret image.
19. $[\text{CAc1, CDc1}] \leftarrow \text{lwt(double(frame), LS)}$ // Except the first frame combine all the audio frames and obtain the IWT to get approximate and detail coefficients. First CD coefficient gives the number of secret coefficients hidden.
20. Extract the third, fourth, and fifth bits of other detail coefficients as per the number of secret bits hidden.
21. Make the bits into 3 groups and perform the majority evaluation and obtain the secret coefficients.
22. Perform inverse IWT considering detailed coefficients as 0 to get the secret audio $S3$.
23. return $S1$, $S2$, and $S3$

Fig. 6.8. VSAVI extracting algorithm
6.2.1.1. Results and Discussion

To measure the video quality, in addition to the PSNR and the SSIM, the VQM is used. The VQM is developed by the Institute for Telecommunication Science (ITS) to provide an objective measurement for the perceived video quality. It measures the perceptual effects of video impairments including blurring, jerky/unnatural motion, global noise, block distortion and color distortion, and combines them into a single metric. The testing results show that the VQM has a high correlation with subjective video quality assessment and has been adopted by American National Standards Institute as an objective video quality standard. The VQM takes the original video and the processed video as input and is computed as follows:

**Calibration:**
This step calibrates the sampled video in preparation for feature extraction. It estimates and corrects the spatial and the temporal shift as well as the contrast and the brightness offset of the processed video sequence with respect to the original video sequence.

**Quality Features Extraction:**
This step extracts a set of quality features that characterizes the perceptual changes in the spatial, temporal, and chrominance properties from spatial-temporal sub-regions of video streams using a mathematical function.

**Quality Parameters Calculation:**
This step computes a set of quality parameters that describe the perceptual changes in video quality by comparing the features extracted from the processed video with those extracted from the original video.

**VQM Calculation**
The VQM is computed using a linear combination of parameters calculated from the previous steps.

\[
VQM = -0.2097 \times si\_loss + 0.5969 \times hv\_loss + 0.2483 \times hv\_gain + 0.0192 \times chroma\_spread - 2.3416 \times si\_gain + 0.0431 \times ct\_ati\_gain + 0.0076 \times chroma\_extreme.
\]

\(si\_loss =\) loss of spatial information (e.g., blurring)
hv_loss = shift of edges from horizontal and vertical orientation to diagonal orientation, such as the case if horizontal and vertical edges suffer more blurring than diagonal edges
hv_gain = shift of edges from diagonal to horizontal and vertical, such as the case if the processed video contains tiling or blocking artifacts
chroma_spread = changes in the spread of the distribution of two-dimensional color samples
si_gain = measures improvements to quality that result from edge sharpening or enhancements.
ct_ati_gain = product of a contrast feature, measuring the amount of spatial detail, and a temporal information feature, measuring the amount of motion present. ct_ati_gain identifies moving-edge impairments that are nearly always present, such as edge noise.

chroma_extreme = detects severe localized color impairments, such as those produced by digital transmission errors

The ITS has developed a video quality metric (BVQM) software tool that performs automated batch processing of multiple video clips to objectively assess their video quality. The BVQM provides a graphical user interface to select the video clips. Processing consists of calibrating the processed video clips and calculating their associated video quality metrics. After processing, the BVQM displays the results graphically and provides text file reports. The BVQM software tool is used to measure the video quality in this research. The cover video is downloaded from https://www.youtube.com/watch?v=3yFiqdCjNMk. One of the frames whose size is 640 x 360 and the secret images are shown in Fig. 6.9. The stego frame and the extracted secret images are shown in Fig. 6.10.
Table 6.3 shows the performance metrics for the stego video, the extracted secret images, and the secret audio. The stego video is evaluated using VQM, which ranges between ‘0’ and ‘1’. Lesser the VQM, better the video quality. When the payload is increased, the quality of the stego is reasonably good even with six secret images and 40000 secret audio samples. However, the quality of the extracted secret audio decreases when the payload is increased further.

<table>
<thead>
<tr>
<th>Cover video</th>
<th>Secret images (grayscale) and Secret audio</th>
<th>Stego</th>
<th>Extracted image</th>
<th>Extracted audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.avi</td>
<td>Two images with size 128 x 128 and 20000 secret audio samples</td>
<td>0.0001</td>
<td>S1:36.2 S2:36.1</td>
<td>S1:0.9949 S2:0.9053</td>
</tr>
<tr>
<td></td>
<td>Four images with size 128 x 128 and 32768 secret audio samples</td>
<td>0.0002</td>
<td>S1:36.2 S2:36.1 S3:35.9 S4:32.4</td>
<td>S1:0.9949 S2:0.9053 S3:0.9894 S4:0.8812</td>
</tr>
<tr>
<td></td>
<td>Six images with size 128 x 128 and</td>
<td>0.0041</td>
<td>S1:36.2 S2:36.1</td>
<td>S1:0.9949 S2:0.9053 S3:0.9894</td>
</tr>
</tbody>
</table>
6.2.1.2. Performance Against Attacks

Table 6.4 shows performance against the RST attack. A rotation of three degrees is applied to all the stego frames and then scaled to original size. It is possible to extract the secret image and the audio with reasonable performance metrics. In the literature, it is not found where the secret information is hidden in both video and audio.

Table 6.4. Performance against RST attack (VSAVI)

<table>
<thead>
<tr>
<th>Cover video</th>
<th>Secret images (grayscale)</th>
<th>Stego VQM</th>
<th>Stego PSNR in dB</th>
<th>Stego SSIM</th>
<th>Extracted image VQM</th>
<th>Extracted image PSNR in dB</th>
<th>Extracted image SSIM</th>
<th>Extracted audio SNR in dB</th>
<th>Extracted audio SPCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.avi</td>
<td>Two images with size 128 x 128 and 20000 secret audio samples</td>
<td>0.0591</td>
<td>S1:29</td>
<td>S2:30.5</td>
<td>S1:0.7940</td>
<td>S2:0.8105</td>
<td>29.5</td>
<td>0.7985</td>
<td></td>
</tr>
</tbody>
</table>

6.2.2 MP4 Video Steganography

The MP4 video files are highly compressed files and the header is only one byte [102]. Only a few bits of this byte can be used for steganography. Therefore, header is not a good candidate for information hiding.

Always the first frame in a video is I-frame and the first frame in a GOP is I-frame. Since I-frames are not lost during compression or any kind of signal processing operations, they are selected for embedding. Although neural network or genetic algorithm can be used for the selection of optimal frame depending on the secret information, it will not exclude P- or B-frames. All I-frames can be used for embedding without effecting the visual quality of the video. All audio frames can be used for embedding.
For MP4, typical GOP size is 12. That means every 13th frame is an I-frame. In H.264/AVC, which is the codec for MP4, the approximate quantization parameter for I-frame is 32 and for P and B frames it is 34 and 36 respectively. The theoretical maximum value for any of these frames is 60. Therefore, I-frames are selected and only those pixels with values greater than 70 are selected for steganography, so that the secret data is not lost. In addition, the secret information is hidden in duplicates as it is done in the case of MP3 files.

The secret images are hidden in the video frames and the secret audio signals are hidden in the audio frames. To resist frame loss, the secret information is stored in duplicates. The technique is referred as MP4VS and the algorithms are given in Figs. 6.11 and 6.12. Fig. 6.11a shows the block diagram of the embedding procedure.

<table>
<thead>
<tr>
<th>Algorithm: Embed-(MP4VS). Embeds the secret images and audio in the MP4 video.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>• Cover video C.mp4</td>
</tr>
<tr>
<td>• Secret image S1.jpg and S2.jpg</td>
</tr>
<tr>
<td>• Secret audio S3.wav.</td>
</tr>
<tr>
<td><strong>Output:</strong> Stego video, G.mp4.</td>
</tr>
<tr>
<td><strong>Method:</strong></td>
</tr>
<tr>
<td>1. VideoFreader ( \leftarrow ) vision.VideoFileReader('C.mp4')</td>
</tr>
<tr>
<td>2. vid ( \leftarrow ) VideoReader('C.mp4')</td>
</tr>
<tr>
<td>3. ( n \leftarrow ) vid.NumberOfFrames</td>
</tr>
<tr>
<td>4. for ( i=1 ) to ( n ) do</td>
</tr>
<tr>
<td>5. ([I, Audio] \leftarrow ) step(VideoFreader)</td>
</tr>
<tr>
<td>6. ( a[i] \leftarrow ) Audio</td>
</tr>
<tr>
<td>7. ( im[i] \leftarrow I )</td>
</tr>
</tbody>
</table>
| 8. endfor // Read cover video and get video and audio frames. Store audio frames in ‘a’ and //video frames in ‘im’.
//In every 13th frame, i.e, 1st, 13th, 26th etc., the secret images are hidden three times continuously. The frame is decomposed into Red Green and Blue components. Two secret images are hidden; one in Blue and one in Green components. From these frames, all pixels with values greater than 70 are used for information hiding// |
| 9. \([sLL1, sHL1, sLH1, sHH1] \leftarrow \) lwt2(double(S1), LS) |
| 10. \([sLL2, sHL2, sLH2, sHH2] \leftarrow \) lwt2(double(S2), LS) // Obtain IWT of the secret images. |
| 11. Hide the low frequency sub-band of the secret image S1 and S2, three times by |
modifying fourth, fifth and sixth bits of the pixels of the green component with value more than 70. Also, hide the number of bits of $S1$ and $S2$ hidden.

12. Integrate R, G, and B components into single image frame
13. Add the frame to the video in the correct location.
14. The secret audio is hidden using the same logic as explained in VSAVI technique.
15. Write the video and audio frames into G.mp4.
16. return stego video $G.mp4$

Fig. 6.11. MP4VS embedding algorithm

**Algorithm: Extract-(MP4VS).** Extracts the secret images and secret audio from the stego video.

**Input:** Stego video $G.mp4$

**Output:**
- Secret image $S1.jpg$ and $S2.jpg$
- Secret audio $S3.wav$

**Method:**
1. $VideoFreader \leftarrow vision.VideoFileReader(‘G.mp4’)$
2. $vid \leftarrow VideoReader(‘G.mp4’)$
3. $n \leftarrow vid.NumberOfFrames$
4. for $i=1$ to $n$
5.   $[I,Audio] \leftarrow step(VideoFreader)$
6.   $a1[i] \leftarrow Audio$
7.   $im1[i] \leftarrow I$
8. endfor // Read stego video and get video and audio frames. Store audio frames in ‘$a1$’ and //video frames in ‘$im1$’
9. Decompose every 1$^{st}$, 13$^{th}$, 26$^{th}$, etc. video frames into Blue and Green components.
10. Make the bits into three groups and perform the majority evaluation and obtain the secret coefficients
11. $S1 \leftarrow ilwt2(newSLL1, 0, 0, 0, LS)$
12. $S2 \leftarrow ilwt2(newSLL2, 0, 0, 0, LS)$ // Obtain the inverse IWT by considering zeroes
13. for high frequency components to get secret images.
14. The audio is extracted in the same way as that explained in VSAVI.
15. return $S1.jpg$, $S2.jpg$, and $S3.wav$

Fig. 6.12. MP4VS extracting algorithm
Fig. 6.11a. Block diagram – MP4VS (Embedding)
6.2.2.1. Results and Discussion

The video source for testing the technique is downloaded from https://www.youtube.com/watch?v=3yFiqdCjNMk. Fig. 6.13 shows one of the frames whose size is 640 x 360. Images of Football and Earth shown in Fig. 6.9 are used as secret images. The stego frame and the extracted secret images are shown in Fig. 6.14. Table 6.5 shows the performance metrics for the stego video and extracted secret images and secret audio. The payload capacity depends on the cover video because only those pixel values or samples with values above the threshold are considered for embedding. C.mp4 is approximately a two-minute video with a frame rate of 30. With this cover video, it is possible to hide up to four images of size 128 x 128 and 32768 audio samples. With this payload, the VQM for the stego video is reasonable and the secret images and the audio samples can be extracted without much distortion.

![Fig. 6.13. One of the frames from the video (MP4VS)](image1.jpg)

![Fig. 6.14. Stego frame and Extracted images (MP4VS): (a) stego frame (b) extracted football (c) extracted earth](image2.jpg)

Table 6.5. Performance metrics for MP4 video steganography (MP4VS)
### Table 6.6. Performance against RST attack (MP4VS)

<table>
<thead>
<tr>
<th>Cover video</th>
<th>Secret images (grayscale)</th>
<th>Stego</th>
<th>Extracted image</th>
<th>Extracted audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.mp4</td>
<td>Two images with size 128 x 128 and 20000 secret audio samples</td>
<td>0.0003</td>
<td>S1:31, S2:32</td>
<td>S1:0.9023, S2:0.9565</td>
</tr>
<tr>
<td></td>
<td>Four images with size 128 x 128 and 32768 secret audio samples</td>
<td>0.0052</td>
<td>S1:30.5, S2:29, S3:30, S4:28</td>
<td>S1:0.9015, S2:0.8953, S3:0.9025, S4:0.8825</td>
</tr>
</tbody>
</table>

#### 6.2.2.2. Performance Against Attacks

Table 6.6 shows performance against RST attack. It is possible to extract the secret image and the audio. A rotation of two degrees is applied to all the stego frames and then scaled to original size. It is possible to extract the secret image and the audio with reasonable performance metrics. If the rotation is more than this, the secret data cannot be retrieved.

Table 6.7 shows the comparison of proposed video steganography techniques with the other video steganography researches. The comparison is done in terms of PSNR with reference to the payload capacity percentage. The proposed technique, VSIWT - IIV is not an efficient technique, since it cannot resist attacks. The proposed VSAVI technique is compared with the works of C. Ozdemir et al. [69] and Y. Yuanzhi et al. [84]. In these two techniques secret message bits are hidden...
in the AVI video frames in spatial domain. C. Ozdemir et al.’s technique gives better capacity and security than the proposed technique. Here the capacity is more by 7% and PSNR is more by 14dB. Compared to Y. Yuanzhi et al.’s work the capacity is improved by 17.5% in the proposed technique. The proposed MP4VS is compared with S. Po-Chyi et al.’s technique [78] that hides both in the video and audio frames of an FLV video. But the performance of the audio is not analyzed in this paper and hence only PSNR is compared. The proposed technique in this case improves the capacity by 4.5% with a decrease in PSNR by 5dB.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>C. Ozdemir et al. [69]</th>
<th>Y. Yuanzhi et al. [84]</th>
<th>Proposed VSAVI</th>
<th>S. Po-Chyi et al. [78]</th>
<th>Proposed MP4VS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload capacity in %</td>
<td>24.8</td>
<td>0.008</td>
<td>17.6</td>
<td>13.7</td>
<td>18.2</td>
</tr>
<tr>
<td>PSNR in dB</td>
<td>48</td>
<td>34.3</td>
<td>34</td>
<td>36</td>
<td>31</td>
</tr>
</tbody>
</table>

6.3. Summary

In this chapter three video steganography techniques are described. VSIWT-IIIV embeds secret images in MP4 video, in which the stego video should be stored in AVI format to avoid data loss and it cannot resist compression. The VSAVI technique that uses uncompressed AVI files as the cover to hide secret images and audio, cannot resist compression. MP4VS hides secret images and audio in MP4 cover video and in this technique the stego video can be stored in compressed MP4 format. All these techniques are implemented in IWT domain and the analysis is done considering capacity and security. The metrics used are PSNR, CQM, and VQM. A comparison of the related works shows the outperformance of the proposed technique.