CHAPTER 5

IMAGE SCALING

5.1 SCOPE

The chapter discusses the image scaling application required to zoom the image with interpolation. The chapter also proposes the novel approach for image scaling and the proposed method is discussed in detail with mathematical equations. The proposed novel method for image scaling is compared with the other existing popular interpolation techniques. The popular interpolation techniques such as nearest neighbour and bilinear are considered for the comparison with the proposed novel approach. The selection of nearest and bilinear for comparison is due to fast computation compared to other interpolation techniques. The scope of the topic is limited to use the different image scaling techniques to fit the image on the display devices of different resolutions. The proposed method, nearest neighbour and bilinear interpolation techniques is ported on a digital signal processor DM 642 to measure the computation complexity and memory requirements. The chapter also discusses the interpolation techniques with the effect of noise. The different image scaling techniques considered are compared on the basis of objective and subjective measures.
5.2 INTRODUCTION TO IMAGE SCALING

Due to the restriction of resolution or display size of mobile devices, spatially downscaled or down sampled Video/Image is provided for the display. But user may not be satisfied the small sized Video/Image and may wants to see the Video/Image with larger resolution. The better solution to satisfy the viewer is to define the region of interest to view the Video/Image. Even if the display size is same, semantically meaningful region can be defined with better resolution. In most of the Videos/Images, certain region is more important than the other regions in the Video/Image. For example,
people in the picture are more meaningful than background. For this reason, region of interest is one of the requirements and figure 5.1 shows the usage of the region of interest scalability.

The Video/Images, binary images or pseudo-binary images such as documents and signatures are major inputs required to process for mobile vision applications. To process these inputs on mobile devices, computational complexity or computation time and memory required to fit the image scaling techniques become crucial. Hence the fast interpolation techniques which gives reasonably good quality are required to process (scale) these inputs are required. When an image is geometrically transformed for scaling purpose, a pixel in the new image is often projected back to a point with non-integer coordinates in the original image.

Cost analysis is appropriate and important criterion for comparison of different algorithms. If the resources such as data memory, program memory and clock speed are limited in the mobile device, efficient utilization of the resources are required. Depending on the critical resource of the application, equivalent evaluation method should be applied. For real time application, design and evaluation strategy should be based upon minimizing the run time factor. Data memory and computational burden analysis is vital in application that interpolate large spatial resolution images. Mobile phones share a restricted amount of embedded memory for Video/Image processing. The interpolation algorithm must be designed and optimized for Video/Image processing application in order to meet the real time performance restrictions.

To illustrate the necessity of fast algorithms for image scaling, consider the linear interpolation defined with four neighboring points with integer coordinates \((Q_{11}, Q_{12}, Q_{21}, \text{ and } Q_{22})\). To estimate the value of point \(P\) from its four neighboring points with coordinates \((Q_{11}, Q_{12}, Q_{21}, \text{ and } Q_{22})\), the linear interpolation in X direction is performed to obtain \(f(x,y)\) which is given by
\[
\begin{align*}
f(x, y) & \approx \\
& \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \\
& \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \\
& \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\
& + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1) 
\end{align*}
\] (5.1)

From the equation (5.1), we can estimate how many floating point multiplication and addition operations are required to find \( f(x, y) \). For linear interpolation with \( x_2 - x_1 = 1 \) and \( y_2 - y_1 = 1 \), we need to calculate \( f(Q_{11})(x_2-x)(y_2-y) \), \( f(Q_{21})(x-x_1)(y_2-y) \), \( f(Q_{12})(x_2-x)(y-y_1) \) and \( f(Q_{22})(x-x_1)(y-y_1) \), each of which requires two floating point subtraction and multiplication operations. Therefore, each pixel in the interpolated image will require \((2 \times 4 + 3 = 11)\) floating point additions (subtractions), and \((2 \times 4 = 8)\) floating point multiplications. Hence the interpolation of an image at VGA (640 X 480) resolution requires \( 640 \times 480 \times 11 = 3,379,200 \) floating point additions, and \( 640 \times 480 \times 8 = 2,457,600 \) floating point multiplications. If the algorithm is tested for interpolation of an image of size 320 X 240 (Source Input Format) to VGA resolution on hardware, the process takes more time. The reason of the slow speed is that mobile devices often use software emulation to process floating point calculations instead of using a specific hardware floating point processor as is typical on a PC. Hence it is required to reduce the computational complexity to fit the algorithm for Video/Image scaling. Also the mobile devices have limited memory and it is required to estimate the memory requirements. Hence it is required to use the efficient image scaling algorithm for the mobile applications which is more efficient in-terms of computational requirements and memory requirements compared to the existing algorithms for image scaling.

5.3 METHODOLOGY USED FOR IMAGE SCALING

In the event that H.264 is decoding QCIF resolution images and it is required to display the images for different resolutions. The scaling algorithm is required to enlarge
the image with acceptable image quality. The decompressed images are of QCIF format with 176 X 144 resolutions. To fit the decoded frames on a mobile device, scaling algorithm is essential. The novel approach for image scaling using interpolation is discussed and algorithm is compared with the other popular algorithms such as nearest
neighbour and bilinear interpolation techniques. The proposed algorithm is compared by considering different comparison parameters. It is possible to scale the image to required resolution for mobile applications. Algorithm is proposed to scale the image to fit the Video/Image on the display devices of any resolutions such as VGA, HDTV (High definition television), etc.

The figure 5.2 and 5.3 shows the novel method proposed for image scaling. The figure 5.2 shows the block operation on the input original image with block size $b_{L1} \times b_{L2}$ and figure 5.3 shows the block operation of the scaled image after interpolation using proposed approach. Proposed approach is based on replacing every $b_{L1} \times b_{L2}$ blocks in the original image by $B_{L1} \times B_{L2}$ blocks where $b_{L1}, b_{L2}$ are the block sizes of the original image and $B_{L1}, B_{L2}$ are the block sizes of the scaled image. The dimension of the scaled image after interpolation is $(m \times \frac{B_{L1}}{b_{L1}} \times n \times \frac{B_{L2}}{b_{L2}})$. The different block sizes can be considered for image scaling. The proposed method discusses the different block sizes with computational complexity. Computational complexity is discusses in terms of number of substitution operation, multiplications and additions.

For an image of dimension $m \times n$ and block size $b_{L1} \times b_{L2}$, the number of blocks are $\frac{m}{b_{L1}}$ and $\frac{n}{b_{L2}}$. For a block size $b_{L1} \times b_{L2}$ in the original image and block size $B_{L1} \times B_{L2}$ in the scaled image after interpolation the number of substitution operations are $(O_1 \times \frac{m}{b_{L1}} \times \frac{n}{B_{L2}})$, the number of multiplication operations are $(O_2 \times \frac{m}{b_{L1}} \times \frac{n}{B_{L2}})$ and number of addition operations are $(O_3 \times \frac{m}{B_{L1}} \times \frac{n}{B_{L2}})$ where $O_1$, $O_2$, $O_3$ are the number of substitution operation, multiplication operation and addition operation for each block. The different cases are considered for the proposed method to estimate the computational complexity in terms of number of substitution operation, multiplication and additions.
**CASE 1:**

Let the dimension of the block size of the original image is $2 \times 2$ ($b_{11} \times b_{12}$)

Let the dimension of the block size of the scaled image $3 \times 3$ ($B_{11} \times B_{12}$)

Scaling factor is $(\frac{B_{11}}{b_{11}} \times \frac{B_{12}}{b_{12}}) = (1.5 \times 1.5)$

![Figure 5.4: Illustration of block operation to obtain new pixels for interpolated image](image)

Considering the $i^{th}$ pixel in the original image block $O$ as $O_i$ for $i=1$ to 4 and $j^{th}$ pixels in the scaled image block $N$ as $N_j$ for $j=1$ to 4. The edge pixel values of the scaled image block is same as that of original image block as shown in figure 5.4 which is given by

$$N_1 = O_1, \quad N_3 = O_2, \quad N_7 = O_3 \quad \text{and} \quad N_9 = O_4$$  \hspace{1cm} (5.2)

The remaining new pixels $N_2, N_4, N_5, N_6, N_8$ can be obtained from the pixels $O_i$ for $i=1$ to 4 and the weight factor. Let $a_1, a_2, a_3$ and $a_4$ be the weight factors associated with the pixels $O_1, O_2, O_3, O_4$ respectively.

$$a_1 \times O_1 + a_2 \times O_2 = N_2$$  \hspace{1cm} (5.3)

$$a_1 \times O_1 + a_3 \times O_3 = N_4$$  \hspace{1cm} (5.4)

$$0.5 \times (a_1 \times O_1 + a_2 \times O_2 + a_3 \times O_3 + a_4 \times O_4) = N_5$$  \hspace{1cm} (5.5)

$$a_2 \times O_2 + a_4 \times O_4 = N_6$$  \hspace{1cm} (5.6)

$$a_3 \times O_3 + a_4 \times O_4 = N_8$$  \hspace{1cm} (5.7)

The weight factors $a_1, a_2, a_3$ and $a_4$ are computed as follows.

$$a_1 + a_2 = 1 \quad a_1 + a_3 = 1 \quad a_2 + a_4 = 1 \quad a_3 + a_4 = 1$$

$$a_1 + a_2 + a_3 + a_4 = 2$$

Simplifying the equations we get $a_1 = a_4$ and $a_2 = a_3$

Hence the equation 5.3 to 5.7 becomes
From the equations 5.2, 5.8 to 5.12,
For 2 X 2 to 3 X 3 the number of new pixels is 9.
4 substitution operations for evaluating $N_1$, $N_3$, $N_7$ and $N_9$
9 multiplications and 5 additions operation are required to evaluate $N_2$, $N_4$, $N_5$, $N_6$ and $N_8$.

For a block size of 2 X 2 to 3 X 3, with a dimension of 176 X 144 (QCIF frame),
The number of substitution operations are $(4 \times 176/2 \times 144/2 = 25344)$
The number of multiplications are $(9 \times 176/2 \times 144/2 = 57024)$ and
The number of additions are $(5 \times 176/2 \times 144/2 = 31680)$

**CASE 2:**
Let the dimension of the block size of the original image is 3 X 3 ($b_{l1} \times b_{l2}$)
Let the dimension of the block size of the scaled image 5 X 5 ($B_{l1} \times B_{l2}$)
Scaling factor is $(\frac{B_{l1}}{b_{l1}} \times \frac{B_{l2}}{b_{l2}}) = (1.66 \times 1.66)$

$$O = \begin{bmatrix} O_1 & O_2 & O_3 \\ O_4 & O_5 & O_6 \\ O_7 & O_8 & O_9 \end{bmatrix} \quad N = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 & N_5 \\ N_6 & N_7 & N_8 & N_9 & N_{10} \\ N_{11} & N_{12} & N_{13} & N_{14} & N_{15} \\ N_{16} & N_{17} & N_{18} & N_{19} & N_{20} \\ N_{21} & N_{22} & N_{23} & N_{24} & N_{25} \end{bmatrix}$$

Figure 5.5: Pixels of the blocks of the original image and scaled image (3 X 3 to 5X 5)
Considering the \( i^{th} \) pixel in the original image block \( O \) as \( O_i \) for \( i=1 \) to 9 and \( j^{th} \) pixels in the scaled image block \( N \) as \( N_j \) for \( j=1 \) to 16. The block of the original image and block of the scaled image block is shown in figure 5.5. The alternate rows and columns of the scaled image block is same as the original image block which is given by
\[
N_1 = O_1, \; N_3 = O_2, \; N_5 = O_3, \; N_{11} = O_4, \; N_{13} = O_5, \; N_{15} = O_6, \; N_{21} = O_7, \; N_{23} = O_8, \; N_{25} = O_9
\]  
(5.13)

The new pixels \( N_2, \; N_4, \; N_6, \; N_7, \; N_8, \; N_9, \; N_{10}, \; N_{12}, \; N_{14}, \; N_{16}, \; N_{17}, \; N_{18}, \; N_{19}, \; N_{20}, \; N_{22}, \; N_{24} \) is found from the old pixels. The weight factors \( a_1, \; a_2, \; a_3, \; a_4, \; a_5, \; a_6, \; a_7, \; a_8 \) and \( a_9 \) are defined to obtain new pixels.
\[
a_1 \cdot O_1 + a_2 \cdot O_2 = N_2
\]  
(5.14)
\[
a_2 \cdot O_2 + a_3 \cdot O_3 = N_4
\]  
(5.15)
\[
a_1 \cdot O_1 + a_4 \cdot O_4 = N_6
\]  
(5.16)
\[
a_4 \cdot O_4 + a_5 \cdot O_5 = N_{12}
\]  
(5.17)
\[
a_2 \cdot O_2 + a_5 \cdot O_5 = N_8
\]  
(5.18)
\[
0.5 \cdot (a_1 \cdot O_1 + a_2 \cdot O_2 + a_4 \cdot O_4 + a_5 \cdot O_5) = N_7
\]  
(5.19)
\[
a_5 \cdot O_5 + a_6 \cdot O_6 = N_{14}
\]  
(5.20)
\[
a_3 \cdot O_3 + a_6 \cdot O_6 = N_{10}
\]  
(5.21)
\[
0.5 \cdot (a_2 \cdot O_2 + a_3 \cdot O_3 + a_5 \cdot O_5 + a_6 \cdot O_6) = N_9
\]  
(5.22)
\[
a_4 \cdot O_4 + a_7 \cdot O_7 = N_{16}
\]  
(5.23)
\[
a_5 \cdot O_5 + a_8 \cdot O_8 = N_{18}
\]  
(5.24)
\[
a_7 \cdot O_7 + a_8 \cdot O_8 = N_{22}
\]  
(5.25)
\[
0.5 \cdot (a_4 \cdot O_4 + a_5 \cdot O_5 + a_7 \cdot O_7 + a_8 \cdot O_8) = N_{17}
\]  
(5.26)
\[
a_6 \cdot O_6 + a_9 \cdot O_9 = N_{20}
\]  
(5.27)
\[
a_8 \cdot O_8 + a_9 \cdot O_9 = N_{24}
\]  
(5.28)
\[
0.5 \cdot (a_5 \cdot O_5 + a_6 \cdot O_6 + a_8 \cdot O_8 + a_9 \cdot O_9) = N_{19}
\]  
(5.29)

The weight facors \( a_1, \; a_2, \; a_3, \; a_4, \; a_5, \; a_6, \; a_7, \; a_8 \) and \( a_9 \) are computed as follows.
\[
a_1 + a_2 = 1 \quad a_1 + a_4 = 1 \quad a_2 + a_3 = 1 \quad a_2 + a_5 = 1
\]
\[ a_3 + a_6 = 1 \quad a_4 + a_5 = 1 \quad a_4 + a_7 = 1 \quad a_5 + a_8 = 1 \]
\[ a_5 + a_6 = 1 \quad a_6 + a_9 = 1 \quad a_8 + a_9 = 1 \]
\[ 0.5(a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9) = 2 \]
Simplifying the equations we get \( a_1 = a_6 = a_8 \quad a_2 = a_4 = a_9 \quad a_3 = a_5 = a_7 \)

Hence the equations 5.14 to 5.29 become

\[ a_1 \times O_1 + a_2 \times O_2 = N_2 \quad (5.30) \]
\[ a_2 \times O_2 + a_3 \times O_3 = N_4 \quad (5.31) \]
\[ a_1 \times O_1 + a_2 \times O_4 = N_6 \quad (5.32) \]
\[ a_2 \times O_4 + a_3 \times O_5 = N_{12} \quad (5.33) \]
\[ a_2 \times O_2 + a_3 \times O_5 = N_6 \quad (5.34) \]
\[ 0.5(a_1 \times O_1 + a_2 \times O_2 + a_2 \times O_4 + a_3 \times O_5) = N_7 \quad (5.35) \]
\[ a_3 \times O_5 + a_1 \times O_6 = N_{14} \quad (5.36) \]
\[ a_3 \times O_3 + a_1 \times O_6 = N_{10} \quad (5.37) \]
\[ 0.5(a_2 \times O_2 + a_3 \times O_3 + a_3 \times O_5 + a_1 \times O_6) = N_9 \quad (5.38) \]
\[ a_2 \times O_4 + a_3 \times O_7 = N_{16} \quad (5.39) \]
\[ a_3 \times O_5 + a_1 \times O_8 = N_{18} \quad (5.40) \]
\[ a_3 \times O_7 + a_1 \times O_8 = N_{22} \quad (5.41) \]
\[ 0.5(a_2 \times O_4 + a_3 \times O_5 + a_3 \times O_7 + a_1 \times O_8) = N_{17} \quad (5.42) \]
\[ a_1 \times O_6 + a_2 \times O_9 = N_{20} \quad (5.43) \]
\[ a_1 \times O_8 + a_2 \times O_9 = N_{24} \quad (5.44) \]
\[ 0.5(a_3 \times O_5 + a_1 \times O_6 + a_1 \times O_8 + a_2 \times O_9) = N_{19} \quad (5.45) \]

From the equations 5.13 and 5.30 to 5.45,
The number of new pixels is 25
9 substitution operations are involved to obtain \( N_1, N_3, N_5, N_{11}, N_{13}, N_{15}, N_{21}, N_{23} \) and \( N_{25} \)
44 multiplications and 24 additions are involved to obtain new pixels.
For a block size of 3 X 3 to 5 X 5, with a dimension of 176 X 144,
The number of substitution operations are \((9 \cdot 177/3 \cdot 144/3 = 25488)\)
The number of multiplications are \((44 \cdot 177/3 \cdot 144/3 = 124608)\) and
The number of additions are \((24 \cdot 177/3 \cdot 144/3 = 67968)\)

**CASE 3:**
Let the dimension of the block size of the original image is 2 X 2 \((b_{l1} \times b_{l2})\)
Let the dimension of the block size of the scaled image 4 X 4 \((B_{l1} \times B_{l2})\)
Scaling factor is \((\frac{B_{l1}}{b_{l1}} \times \frac{B_{l2}}{b_{l2}}) = (2 \times 2)\)

\[
O = \begin{bmatrix}
O_1 \\
O_2 \\
O_3 \\
O_4 \\
\end{bmatrix}
N = \begin{bmatrix}
N_1 & N_2 & N_3 & N_4 \\
N_5 & N_6 & N_7 & N_8 \\
N_9 & N_{10} & N_{11} & N_{12} \\
N_{13} & N_{14} & N_{15} & N_{16} \\
\end{bmatrix}
\]

**Figure 5.6:** Pixels of the blocks of the original image and scaled image (2 X 2 to 4 X 4)

Considering the \(i^{th}\) pixel in the original image block \(O\) as \(O_i\) for \(i=1\) to 9 and \(j^{th}\) pixels in the scaled image block \(N\) as \(N_j\) for \(j=1\) to 16. The block of the original image and block of the scaled image block is shown in figure 5.6. The alternate rows and columns of the scaled image block is same as the original image block which is given by

\[N_1 = O_1, N_3 = O_2 = N_4, N_5 = O_3, N_{11} = O_4 = N_{12} = N_{11} = N_{15} = N_{16}\] (5.46)

The new pixels \(N_2, N_4, N_5, N_6, N_7, N_8, N_{10}, N_{12}, N_{13}, N_{14}, N_{15}, N_{16}\) is found from old pixels. The weight factors \(a_1, a_2, a_3\) and \(a_4\) are defined to measure the new pixels.

\[a_1 \cdot O_1 + a_2 \cdot O_2 = N_2\] (5.47)
\[a_1 \cdot O_1 + a_3 \cdot O_3 = N_5\] (5.48)
\[a_2 \cdot O_2 + a_4 \cdot O_4 = N_7 = N_8\] (5.49)
\[a_3 \cdot O_3 + a_4 \cdot O_4 = N_{10} = N_{14}\] (5.50)
\[0.5 \cdot (a_1 \cdot O_1 + a_2 \cdot O_2 + a_3 \cdot O_3 + a_4 \cdot O_4) = N_6\] (5.51)
The weight facors $a_1$, $a_2$, $a_3$ and $a_4$ are computed as follows.

\begin{align*}
a_1 + a_2 &= 1 \\
a_1 + a_3 &= 1 \\
a_2 + a_4 &= 1 \\
a_3 + a_4 &= 1 \\
a_1 + a_2 + a_3 + a_4 &= 2
\end{align*}

Simplifying the equations we get $a_1 = a_4$ and $a_2 = a_3$.

Hence the equations 5.47 to 5.51 become

\begin{align*}
a_1 \cdot O_1 + a_2 \cdot O_2 &= N_2 \\
a_1 \cdot O_1 + a_2 \cdot O_3 &= N_5 \\
a_2 \cdot O_2 + a_1 \cdot O_4 &= N_7 = N_8 \\
a_2 \cdot O_3 + a_1 \cdot O_4 &= N_{10} = N_{14} \\
0.5 \cdot (a_1 \cdot O_1 + a_2 \cdot O_2 + a_2 \cdot O_3 + a_1 \cdot O_4) &= N_6
\end{align*}

From the equations 5.46 and 5.52 to 5.56,

The numbers of new pixels are 16

9 substitution operations for finding the new pixels $N_1, N_3, N_4, N_8, N_{12}, N_{13}, N_{14}, N_{15}, N_{16}$

13 multiplication operations and 7 addition operations are involved.

For a block size of 2 X 2 to 4 X 4, with a dimension of the image 176 X 144,

The number of substitution operations are $(9 \cdot 176/2 \cdot 144/2 = 57024)$

The number of multiplications are $(13 \cdot 176/2 \cdot 144/2 = 82368)$ and

The number of additions are $(7 \cdot 176/2 \cdot 144/2 = 44352)$

Table 5.1: Computational complexity for different block sizes for the proposed approach

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Block size & Block size & Substitution & Multiplication & Addition \\
$\mathbf{b_L_1}$ X $\mathbf{b_L_2}$ & $\mathbf{B_L_1}$ X $\mathbf{B_L_2}$ & Operations & Operations & Operations \\
\hline
2 X 2 & 3 X 3 & 4 & 9 & 5 \\
\hline
3 X 3 & 5 X 5 & 9 & 44 & 24 \\
\hline
2 X 2 & 4 X 4 & 9 & 13 & 7 \\
\hline
\end{tabular}
\caption{Computational complexity for different block sizes for the proposed approach}
\end{table}
Table 5.2: Computational complexity for different block sizes for the proposed approach
(For QCIF frame)

<table>
<thead>
<tr>
<th>Block size b_{L1} X b_{L2}</th>
<th>Block size B_{L1} X B_{L2}</th>
<th>Substitution Operations</th>
<th>Multiplication Operations</th>
<th>Addition Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 X 2</td>
<td>3 X 3</td>
<td>25344</td>
<td>57024</td>
<td>31680</td>
</tr>
<tr>
<td>3 X 3</td>
<td>5 X 5</td>
<td>25488</td>
<td>124608</td>
<td>67968</td>
</tr>
<tr>
<td>2 X 2</td>
<td>4 X 4</td>
<td>57024</td>
<td>82368</td>
<td>44352</td>
</tr>
</tbody>
</table>

The computational complexity for the different block sizes are shown in the table 5.1 and table 5.2. From table 5.1 and 5.2, it is clear that the number of substitution operations, multiplication and addition operations required are less for block size 2 X 2 to 3 X 3 compared to the other blocks shown in the table 5.1 and 5.2. Hence the selection of 2 X 2 blocks in the original image reduces the computational overhead which is the key parameters for consideration in mobile applications. Block size 3 X 3 to 4 X 4 is not considered as it gives less scaling factor considered to block size 2 X 2 to 3 X 3 but it provides good quality. The block size 2 X 2 to 4 X 4 provides high scaling factor but it provides low quality. Hence it is not considered for the given mobile applications. The proposed approach for block selection is a tradeoff between the computation complexity, scaling factor and the quality in terms of PSNR and quality index. Hence a block size 2 X 2 to 3 X 3 is considered in the proposed method for image scaling. Also, the input considered is of QCIF resolution (176 X 144) and the block operation of block size of 16, 32 and 64 are considered in the previous chapter image compression to apply MSVD. Hence the block size of 2 X 2 to 3 X 3 is preferred as 2 X 2 divides the block size 16, 32 and 64.

A good interpolation algorithm must use less number of memory accesses to reduce the power consumption. Since the algorithm is targeted for mobile application, power consumption is a critical factor in deciding the algorithm. The data memory required in the proposed approach is comparatively less compared to nearest neighbor
and bilinear interpolation. The nearest and bilinear interpolation involves convolution operation for image up scaling. The data memory requirements are strictly connected to the contributing input pixels. The proposed approach uses less data memory compared to the nearest and bilinear interpolation.

5.4 IMPLEMENTATION DETAILS

Different image file formats of different resolutions are considered for the experimentation. The different file formats such as TIFF, JPEG, PNG, BMP, etc are considered for experimentation. Extensive experimentation is done using different file formats of different resolutions and on the different interpolation algorithms such as nearest neighbor and bilinear. The figure 5.7 shows some of the inputs considered for experimentation. The table 5.3 shows some of the inputs considered for the experimentation. Exhaustive experiments are conducted using the proposed method and
the comparison of the result is done with nearest and bilinear interpolation techniques using different comparison parameters. The different comparison parameters considered are MSE, PSNR, quality index, average difference, maximum difference, structural content, normalized absolute error and normalized cross correlation. Also, the algorithms are compared by estimating the memory required to fit the code and computation time required for computation.

5.4.1 Mean Square Error (MSE)

The Mean Square Error measures the difference between the frames which is usually applied to Human Visual System. It is based on pixel-pixel comparison of the image frames. Minimizing the MSE is equivalent to least-squares optimization in a minimum energy sense, for which many mathematical tools are available. MSE is still popular despite its inability to reliably predict perceived quality across different scenes and distortion types.

\[
d(X, Y) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (X_{i,j} - Y_{i,j})^2}{mn}
\]

(5.57)

5.4.2 Peak Signal to Noise Ratio (PSNR)

PSNR is measured on a logarithmic scale and depends on the mean squared error (MSE) of between an original and an impaired image or video frame, relative to \((2^n - 1)^2\) (the square of the highest-possible signal value in the image, where \(n\) is the number of bits per image sample).

\[
PSNR_{dB} = 10 \log_{10} \left( \frac{(2^n - 1)^2}{MSE} \right)
\]

(5.58)

PSNR can be calculated easily and quickly and is therefore a very popular quality measure, widely used to compare the ‘quality’ of images.
5.4.3 Quality Index (QI)

Image quality index measurement does not depend on the image being tested, the viewing conditions or the individual observers. More importantly it must be applicable to various image processing applications and provide meaningful comparison across different types of image distortions.

\[
Q = \frac{4\sigma_{xy}\bar{y}}{(\sigma_x^2 + \sigma_y^2)(\bar{x}^2 + \bar{y}^2)}
\]  
(5.59)

Where

\[
\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i
\]  
(5.60)

\[
\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N} (x_i - \bar{x})^2, \quad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N} (y_i - \bar{y})^2,
\]  
(5.61)

\[
\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})
\]  
(5.62)

The dynamic range of Q is [-1,1]. The best value of 1 is achieved if and only if \( y_i = x_i \) for all \( i = 1, 2, \ldots, N \).

The lowest value of -1 occurs when \( y_i = 2\bar{x} - x_i \) for all \( i = 1, 2, \ldots, N \).

5.4.4 Average Difference

Average difference is calculated by subtracting the original image with that of reconstructed image, and dividing the sum of the resultant matrix with that of the size of the image.

\[
average\ difference = \frac{\text{error}[i][j]}{MN}
\]  
(5.63)

Where, \( \text{error}[i][j] = a[i][j] - b[i][j] \) \( M, N = \) size of the image.

5.4.5 Maximum Difference

Maximum difference is calculated by subtracting the original image with that of reconstructed image, and taking the maximum value among the obtained result.
maximum difference = \( a[i][j] - b[i][j] \)  \hspace{1cm} (5.64)

Where, \( a[i][j] = \text{original image} \), \( b[i][j] = \text{reconstructed image} \).

Structural Content: Structural content is calculated using following formula,

\[
\text{structural content} = \frac{imga[i][j]}{imgb[i][j]}
\]

Where,

\( imga[i][j] = a[i][j] \times a[i][j] \)  \hspace{1cm} (5.66)

\( imgb[i][j] = b[i][j] \times b[i][j] \)  \hspace{1cm} (5.67)

Where, \( a[i][j] = \text{original image} \), \( b[i][j] = \text{reconstructed image} \).

### 5.4.6 Normalized Absolute Error

Normalized absolute error is calculated using following formula,

\[
\text{normalised absolute error} = \frac{\text{totalerr}}{\text{totalimg}},
\]

\( \text{totalerr} = \text{error}[i][j], \quad \text{totalimg} = a[i][j]. \)

\( \text{error}[i][j] = a[i][j] - b[i][j] \) \hspace{1cm} (5.69)

\( \text{error}[i][j] = \text{abs(error}[i][j]) \) \hspace{1cm} (5.70)

We have to take only absolute value of \( \text{error}[i][j] \).

### 5.4.7 Normalized Cross correlation (NCC)

Normalized cross correlation is a mathematical computation that fulfills an essential role in finding the similarity of images image processing.

Normalized Cross Correlation is calculated using following formula,

\[
\text{NCC} = \frac{\text{totalimgab}}{\text{totalimgaa}}
\]

Where,

\( \text{totalimgab} = (a[i][j] \times i[j]) \) \hspace{1cm} (5.72)

\( \text{totalimgaa} = (a[i][j] \times a[i][j]) \) \hspace{1cm} (5.73)
### 5.5 EXPERIMENTAL RESULTS AND DISCUSSIONS

Table 5.3: Comparison of the proposed technique for different weight factors

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Proposed approach With weights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0.1 &amp; 0.9)</td>
</tr>
<tr>
<td>Memory (H)</td>
<td>1144</td>
</tr>
<tr>
<td>Computation time</td>
<td>0.018565</td>
</tr>
<tr>
<td>MSE</td>
<td>94.9462</td>
</tr>
<tr>
<td>PSNR</td>
<td>28.3560</td>
</tr>
<tr>
<td>Quality index</td>
<td>0.8279</td>
</tr>
<tr>
<td>Maximum difference</td>
<td>109</td>
</tr>
<tr>
<td>Average difference</td>
<td>-0.1449</td>
</tr>
<tr>
<td>Structural content</td>
<td>1.0014</td>
</tr>
<tr>
<td>Normalized absolute error</td>
<td>0.0290</td>
</tr>
<tr>
<td>Normalized cross correlation</td>
<td>0.9966</td>
</tr>
</tbody>
</table>

Experiments are conducted for different weights and for various set of input image and file formats. The sample result is displayed in the table 5.3. The weight factor $a_1=0.4$ and $a_2=0.6$ are considered for the proposed approach and the sample results are displayed in the table 5.3. Experiments are repeated for different weight factors such as (0.1, 0.9), (0.2, 0.8), (0.3, 0.7), (0.6, 0.4), (0.7, 0.3), (0.8, 0.2) and (0.9, 0.1) and the comparison is done using different parameters. The table 5.3 shows the proposed technique result for different weight factors. It is seen that the selection of weight factor
depends on the pixel relationship of the input image. If the pixels are more correlated, the weight factor of (0.5, 0.5) gives better result. For the result displayed the correlation factor is 0.9966 which is almost equal to 1. The computation complexity in clock cycles and memory complexity is listed in the table 5.4 by porting the proposed approach, nearest neighbor, bilinear interpolation algorithm on TIDSP. A profiling result gives the computation complexity and memory complexity. The comparison of the proposed approach is done with the other interpolation methods such as nearest neighbor and bilinear interpolation.

Table 5.4: Comparison of proposed approach with nearest neighbor & bilinear interpolation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nearest neighbor interpolation</th>
<th>Bilinear interpolation</th>
<th>Proposed approach Weights ($a_1$, $a_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.4, 0.6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0.6, 0.4)</td>
</tr>
<tr>
<td>Program Memory</td>
<td>1048</td>
<td>1200</td>
<td>1144</td>
</tr>
<tr>
<td>computation time (sec)</td>
<td>0.0136287</td>
<td>0.856</td>
<td>0.018565</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01856569</td>
</tr>
<tr>
<td>MSE</td>
<td>98.5860</td>
<td>10.4</td>
<td>22.8421</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>22.9372</td>
</tr>
<tr>
<td>PSNR</td>
<td>28.1927</td>
<td>37.97</td>
<td>34.5434</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>34.5254</td>
</tr>
<tr>
<td>Quality index</td>
<td>0.8724</td>
<td>0.999</td>
<td>0.9556</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9544</td>
</tr>
<tr>
<td>Maximum difference</td>
<td>138</td>
<td>39</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>66</td>
</tr>
<tr>
<td>Average difference</td>
<td>-0.0640</td>
<td>-0.0294</td>
<td>-0.0324</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-0.0353</td>
</tr>
<tr>
<td>Structural content</td>
<td>0.9986</td>
<td>1.0028</td>
<td>1.0052</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.0052</td>
</tr>
<tr>
<td>Normalized absolute error</td>
<td>0.0397</td>
<td>0.0171</td>
<td>0.0186</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0187</td>
</tr>
<tr>
<td>Normalized cross correlation</td>
<td>0.9967</td>
<td>0.9982</td>
<td>0.9965</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9965</td>
</tr>
</tbody>
</table>
Table 5.5: Comparison of proposed approach with nearest and bilinear interpolation for block size of 2 X 2 images with scaling factor of 1.5

<table>
<thead>
<tr>
<th>Method</th>
<th>Substitution Operations</th>
<th>Multiplication operation</th>
<th>Addition operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest neighbor</td>
<td>176x 144x1.5 = 38016</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bilinear interpolation</td>
<td>0</td>
<td>176x144x1.5x11=418176</td>
<td>176x144x1.5x8=304128</td>
</tr>
<tr>
<td>Proposed Approach</td>
<td>4* 176/2 144/2=25344</td>
<td>9* 176/2 * 144/2=57024</td>
<td>5* 176/2 * 144/2=31680</td>
</tr>
</tbody>
</table>

The table 5.4 and 5.5 indicates that the proposed technique is better than nearest neighbor in terms of PSNR and Quality index but consumes more computation time than the nearest neighbor technique. The proposed technique is better than the bilinear interpolation in terms of computation time which is illustrated in table 5.5. But the PSNR and quality index is less compared to bilinear interpolation. The proposed algorithm is a tradeoff between the quality and the computation complexity. The figure 5.5 shows the original image and image histogram.

![Figure 5.5: Original image and image histogram](image)

![Histogram for nearest neighbor interpolation](image)

Figure 5.8: Scaled image & image histogram using nearest neighbor for a scaling factor 1.5

Nearest neighbor interpolation

Histogram for nearest neighbor interpolation
The image scaling result and its histogram using nearest neighbor technique for a scaling factor of 1.5 is shown in figure 5.8 to 5.10. The different interpolation techniques such as nearest neighbor, bilinear and proposed approach are ported on DSP under noisy conditions. The different type of noise considered is Gaussian, speckle, Poisson and salt & pepper. The table 5.6 shows the result for different scaling algorithms under the effect...
of noise. The table 5.6 shows that the proposed approach performs better under the effect of noise. The PSNR of the image after interpolation using proposed approach is reasonably good even under the effect of noise which is also substantiated by higher value of quality index.

Table 5.6: Comparison of the scaling algorithms under the effect of different noises

<table>
<thead>
<tr>
<th>Scaling algorithms</th>
<th>Noise added</th>
<th>MSE</th>
<th>PSNR</th>
<th>Quality Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest</td>
<td>Gaussian</td>
<td>1321</td>
<td>16.9205</td>
<td>0.9378</td>
</tr>
<tr>
<td></td>
<td>Speckle</td>
<td>1906</td>
<td>15.3296</td>
<td>0.9453</td>
</tr>
<tr>
<td></td>
<td>Poisson</td>
<td>378.0865</td>
<td>22.3549</td>
<td>0.9020</td>
</tr>
<tr>
<td></td>
<td>Salt &amp; pepper</td>
<td>2159</td>
<td>14.7870</td>
<td>0.9646</td>
</tr>
<tr>
<td>Bilinear</td>
<td>Gaussian</td>
<td>541.0906</td>
<td>20.7981</td>
<td>0.9133</td>
</tr>
<tr>
<td></td>
<td>Speckle</td>
<td>774.0885</td>
<td>19.2429</td>
<td>0.9080</td>
</tr>
<tr>
<td></td>
<td>Poisson</td>
<td>158.6316</td>
<td>26.1269</td>
<td>0.9908</td>
</tr>
<tr>
<td></td>
<td>Salt &amp; pepper</td>
<td>863.4177</td>
<td>18.7686</td>
<td>0.9776</td>
</tr>
<tr>
<td>Proposed approach</td>
<td>Gaussian</td>
<td>1058</td>
<td>17.8839</td>
<td>0.9174</td>
</tr>
<tr>
<td></td>
<td>Speckle</td>
<td>1533</td>
<td>16.2730</td>
<td>0.9363</td>
</tr>
<tr>
<td></td>
<td>Poisson</td>
<td>254.6695</td>
<td>24.0710</td>
<td>0.9688</td>
</tr>
<tr>
<td></td>
<td>Salt &amp; pepper</td>
<td>1759</td>
<td>15.6761</td>
<td>0.9730</td>
</tr>
</tbody>
</table>

5.6 SUMMARY AND CONCLUSIONS

The experimental results shows that the proposed approach works better than the nearest neighbor interpolation in terms of quality of the image (PSNR) and better than the bilinear interpolation in terms of computational complexity (computation time). Also, the proposed approach works better under noisy conditions. The different noises such as Gaussian, speckle, Poisson and salt & pepper noises are considered to know the
susceptibility of the proposed approach. The proposed approach is compared with nearest
eighbor and bilinear interpolation methods.

The proposed approach can be modified to obtain the performance of the nearest
neighbor interpolation and bilinear interpolation. The decompressed image results in the
reconstruction of SVD are generally of QCIF format. The proposed approach can be used
to scale-up the QCIF format or CIF format to any resolution such as QVGA, VGA, etc.
The proposed approach can be used iteratively to convert from the given image/video
format to required display resolution.