TECHNICAL REPORT

ISO/IEC TR 15938-8

First edition
2002-12-15

AMENDMENT 1
2004-11-15

Information technology — Multimedia content description interface —

Part 8: **Extraction and use of MPEG-7 descriptions**

AMENDMENT 1: Extensions of extraction and use of MPEG-7 descriptions

Technologies de l'information — Interface de description du contenu multimédia —

Partie 8: Extraction et utilisation des descriptions MPEG-7

AMENDEMENT 1: Extensions d'extraction et utilisation des descriptions MPEG-7

Reference number
ISO/IEC TR 15938-8:2002/Amd.1:2004(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC TR 15938-8:2002 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

NOTE This document preserves the sectioning of ISO/IEC TR 15938-8:2002. The text and figures given in this document are currently being considered as additions and/or modifications to those corresponding sections in ISO/IEC TR 15938-8:2002.

Information technology — Multimedia content description interface —

Part 8: **Extraction and use of MPEG-7 descriptions**

AMENDMENT 1: Extensions of extraction and use of MPEG-7 descriptions

Add after subclause 5.6:

5.7 GofGopFeature

This datatype is used to describe a certain visual feature representative of a series of video frames or collection of pictures. It is obtained by aggregating the visual descriptors extracted from each video frame or image in the collection.

5.7.1 Feature Extraction

First, the extraction algorithm computes a descriptor of the visual feature for each frame in the sequence or each image in the collection. The extraction is specified in the subclauses corresponding to the descriptor used (e.g. for HomogeneousTexture, subclause 4.3.1.1 is used). Once the values of the frame/image-based descriptors are computed, a instance of GofGopFeature is derived by the aggregation procedure corresponding to the descriptor used; as defined in ISO/IEC 15938-3.

There are three aggregation methods (i.e. Average, Median, SplitMerge) as follows:

- Average:

  Each component of descriptors in the GOF or GOP is summed and then averaged to compose the aggregated description

- Median:

  Each component of descriptors in the GOF or GOP is sorted and then the middle value is selected to compose the aggregated description.

- SplitMerge:

  The DominantColor descriptors from different images are aggregated by merging of the clusters ("Value" elements) of different descriptors based on their proximity in colour space (the clusters within the same descriptor are also included as a special case, although if the extraction algorithm from 4.2.3.1 is followed, their distance will be greater than DISTANCE_MIN specified below). The merging procedure is performed iteratively, starting with the closest pair and repeating until only a small number of combined clusters remains. The outline of this algorithm is as follows:

  closest_distance=0

```
While (number_of_clusters > MAX_NUM_OF_CLUSTERS or

        closest_distance < DISTANCE_MIN) {

        1.  find two closest clusters
        2.  merge these two clusters
}
```

The distance between clusters is defined as the Euclidean distance between cluster centres, DISTANCE_MIN is the same as in 4.2.3.1 and MAX_NUM_OF_CLUSTERS is equal to 8.

Merging of the clusters is performed as follows. The representative colour value for the merged cluster is a weighted average of the colour values of the component clusters, where the weights are the *relative* pixel counts in the clusters.

$$m = w_1 m_1 + w_2 m_2$$

Merging of the colour variances is based on the assumption that each colour component is independent and for each component we assume that we are calculating the variance of a weighted sum of two Gaussian distributions. This leads to the following formula for the variance of the merged cluster σ^2:

$$\sigma^2 = w_1 \sigma_1^2 + w_2 \sigma_2^2 + w_1 w_2 (m_1 - m_2)^2 ,$$

where σ_1^2 , σ_2^2 are the variances of the component clusters, m_1 , m_2 are their means and w_1 , w_2 are

w_1 = W1/(W1+W2), w_2 = W2/(W1+W2)

where W1 and W2 are the unquantised weights for sub-descriptors.

5.7.2 Similarity Matching Criteria

Matching of GofGopFeature is performed using the descriptors' matching function appropriate to the descriptor used. Only GofGopFeature descriptors characterizing the same feature can be compared. For example, GofGopFeature using the HomogeneousTexture descriptor for two different sequences can be compared. Some descriptors allow multiple aggregation methods, for example, the Color Layout or Edge Histogram descriptors. Matching of GofGopFeature describing the same feature but derived with a different aggregation method is possible.

5.7.3 DDL instantiation examples

In the following two examples, an instance of ColorLayout is embedded in the GofGopFeature datatype.

In the first example, there is no specification of aggregation method.

```
<GofGopFeature>
   <Descriptor xsi:type="mpeg7:ColorLayoutType">
      <YDCCoeff>48</YDCCoeff>
      <CbDCCoeff>34</CbDCCoeff>
      <CrDCCoeff>32</CrDCCoeff>
      <YACCoeff5>12 10 13 9 10</YACCoeff5>
      <CbACCoeff2>14 15</CbACCoeff2>
      <CrACCoeff2>16 12</CrACCoeff2>
   </Descriptor>
</GofGopFeature>
```


In the second example, “Average” is used to aggregate descriptions.

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:ColorLayoutType” aggregation=”Average”>
      <YDCCoeff>48</YDCCoeff>
      <CbDCCoeff>34</CbDCCoeff>
      <CrDCCoeff>32</CrDCCoeff>
      <YACCoeff5>15 11 13 9 8</YACCoeff5>
      <CbACCoeff2>14 15</CbACCoeff2>
      <CrACCoeff2>16 12</CrACCoeff2>
   </Descriptor>
</GofGopFeature>
```

In the following example, an instance of DominantColor is embedded in the GofGopFeature datatype.

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:DominantColorType” aggregation=”SplitMerge”>
      <SpatialCoherency>0</SpatialCoherency>
      <Value>
         <Percentage>5</Percentage>
         <Index>0 89 203</Index>
         <ColorVariance>0 1 1</ColorVariance>
      </Value>
      <Value>
         <Percentage>14</Percentage>
         <Index>120 43 74</Index>
         <ColorVariance>0 1 0</ColorVariance>
      </Value>
      <Value>
         <Percentage>12</Percentage>
         <Index>243 212 27</Index>
         <ColorVariance>1 0 0</ColorVariance>
      </Value>
   </Descriptor>
</GofGopFeature>
```


In the following two examples, an instance of EdgeHistogram is embedded in the GofGopFeature datatype.

In the first example, there is no specification of aggregation method

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:EdgeHistogramType”>
      <BinCounts>
        2  6  4  4  2  1  7  5  3  2  1  6  4  2  2  2  5  4
        5  3  1  5  5  6  5  2  6  5  4  4  1  6  4  4  4  0  6  3  5
        2  1  5  5  6  6  4  2  3  6  7  3  2  5  5  7  3  2  4  4  7
        1  5  6  4  6  1  5  7  4  5  1  6  4  6  5  1  3  4  7  6
      </BinCounts>
   </Descriptor>
</GofGopFeature>
```

In the second example, “Average” is used to aggregate descriptions.

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:EdgeHistogramType” aggregation=”Average”>
      <BinCounts>
        2  6  4  4  2  1  7  5  3  2  1  6  4  2  2  2  5  4  5  3  1
          5  5  6  5  2  6  5  4  4  1  6  4  4  4  0  6  3  5  2  1  5
          5  6  6  4  2  3  6  7  3  2  5  5  7  3  2  4  4  7  1  5  6
          4  6  1  5  7  4  5  1  6  4  6  5  1  3  4  7  6
      </BinCounts>
   </Descriptor>
</GofGopFeature>
```

In the following two examples, an instance of HomogeneousTexture is embedded in the GofGopFeature datatype.

In the first example, there is no specification of aggregation method.

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:HomogeneousTextureType”>
      <Average>19</Average>
      <StandardDeviation>20</StandardDeviation>
      <Energy>
      103  87  99  130  97  73  112  109  122  132  108  102  105  113
        106  141  103  111  78  76  82  117  88  70  69  61  48 68  48
        53
      </Energy>
      <EnergyDeviation>
      106  84  94  130  94  75  107  104  117  128 100  99  97  107  92
        132  90  106  76  64  78  110  83  65  64 52  39  72  35  47
      </EnergyDeviation>
   </Descriptor>
</GofGopFeature>
```


In the second example, “Median” is used to aggregate descriptions.

```
<GofGopFeature>
   <Descriptor xsi:type=”mpeg7:HomogeneousTextureType” aggregation = ”Median” >
      <Average>19</Average>
      <StandardDeviation>20</StandardDeviation>
      <Energy>
      103  87  99  130  97  73  112  109  122  132  108  102  105  113
        106  141  103  111  78  76  82  117  88  70  69  61  48  68  48
        53
      </Energy>
      <EnergyDeviation>
      106  84  94  130  94  75  107  104  117  128  100  99  97  107 92
        132  90  106  76  64  78  110  83  65  64  52  39  72  35  47
      </EnergyDeviation>
   </Descriptor>
</GofGopFeature>
```

5.7.3 Conditions of Usage

There are no specific conditions and limitations on the use of this container datatype.

Add after subclause 6.8:

6.9 Color Temperature

The color temperature of an image specifies the color of illumination in the scene of the image. It is expressed by Kelvin (K) temperature scale in the [1667K, 25000K] range. Using this, the color temperature descriptor describes the perceptual temperature feeling of an image. It targets the perception-based image browsing that enables viewers to navigate and match images based on the temperature perception (i.e. hot, warm, moderate, and cool) of the image.

This descriptor is also useful when a user would like to change the illumination of scene (i.e. still images or video) in favor of the user's preference. For example, some people might want to see warmer images (e.g. taken under incandescent lights) than original images while some people might want to see cooler images (e.g. taken under bright daylights). Those effects can be automatically achieved by adjusting the color temperature.

6.9.1 Color Temperature Browsing

6.9.1.1 Feature Extraction

The (correlated) color temperature of the scene-illumination in the image is extracted as follows.

Note: In this section, several references are made to sRGB, perceived illuminant, and (correlated) color temperature and its reciprocal scale. All information on these subjects can be found in [AMD1-1][AMD1-2][AMD1-3][AMD1-4][AMD1-5].

6.9.1.1.1 The Overall View of Color Temperature Extraction Algorithm

1) Linearizing input image: $RGB \rightarrow R_lG_lB_l$

2) Converting $R_lG_lB_l$ into XYZ

3) Removing pixels that have the pixel value smaller than the low luminance threshold(T_{ll})

4) Averaging XYZ value for all remained pixels: $X_aY_aZ_a$

5) Calculating the self-luminous threshold: $X_{T_s}, Y_{T_s}, Z_{T_s}$ If $X_{T_s}, Y_{T_s}, Z_{T_s}$ have the same values with the previous values, go to procedure 7), else remove pixels that have the pixel value bigger than the self-luminous threshold and repeat procedure 4) to 6)

6) Averaging XYZ value for all pixels remained, estimating it as the illuminant tri-stimulus values, and computing the scene-illuminant chromaticity coordinates (x_s, y_s) in CIE 1931 diagram

7) Converting the scene-illuminant chromaticity (x_s, y_s) into color temperature T_c

   (1) Calculating the chromaticity coordinates (u_s, v_s) in CIE 1960 UCS diagram from (x_s, y_s)

   (2) Finding two adjacent isotemperature lines from (u_s, v_s) and obtaining the distance from those lines

   (3) Computing the correlated color temperature using the distance ratio

6.9.1.1.2 The Detail of Extraction Algorithm

1) Linearizing input image: Obtain the linearized $R_lG_lB_l$ from the inverse gamma correction of the input RGB, which is the gamma-corrected for display devices

Note, it is assumed that an input image RGB is a gamma-corrected non-linear sR'G'B' in the range of 0~255(8bit) in the following equations.

if $R'_{sRGB}(i,j), G'_{sRGB}(i,j), B'_{sRGB}(i,j) \le 0.03928 \times 255.0$,

$$R_{sRGB}(i,j) = \left(R'_{sRGB}(i,j) / 255 \right) \div 12.92$$

$$G_{sRGB}(i,j) = \left(G'_{sRGB}(i,j) / 255 \right) \div 12.92,$$

$$B_{sRGB}(i,j) = \left(B'_{sRGB}(i,j) / 255 \right) \div 12.92$$

else $R'_{sRGB}(i,j), G'_{sRGB}(i,j), B'_{sRGB}(i,j) > 0.03928 \times 255.0$,

$$R_l(i,j) = R_{sRGB}(i,j) = \left[\frac{\left(R'_{sRGB}(i,j) / 255 \right) + 0.055}{1.055} \right]^{2.4}$$

$$G_l(i,j) = G_{sRGB}(i,j) = \left[\frac{\left(G'_{sRGB}(i,j) / 255 \right) + 0.055}{1.055} \right]^{2.4},$$

$$B_l(i,j) = B_{sRGB}(i,j) = \left[\frac{\left(B'_{sRGB}(i,j) / 255 \right) + 0.055}{1.055} \right]^{2.4}$$

where *(i,j)* is the index for pixels

2) Converting linearized $R_l G_l B_l$ into CIE 1931 tristimulus *XYZ* with conversion matrix *M*

$$\begin{bmatrix} X(i,j) \\ Y(i,j) \\ Z(i,j) \end{bmatrix} = \mathrm{M} \bullet \begin{bmatrix} R_l(i,j) \\ G_l(i,j) \\ B_l(i,j) \end{bmatrix},$$

where conversion matrix $M = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}.$

3) Removing pixels that have the pixel value smaller than the low luminance threshold(T_{ll})

$$\begin{cases} Y(i,j) < T_{ll}, & p(i,j) = 0 \\ otherwise, & p(i,j) = 255 \end{cases},$$

where *p(i, j)* is the label for each pixel at the location (*i*, *j*).

4) Averaging *XYZ* value for all pixels remained, which have *p(i, j) = 255* : $X_aY_aZ_a$. *row * col* intuitively means the number of all pixels remained.

$$X_a = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} X(i,j),$$

$$Y_a = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} Y(i,j),$$

$$Z_a = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} Z(i,j).$$

Calculating the self-luminous threshold: $X_{T_s}, Y_{T_s}, Z_{T_s}$

$$X_{T_s} = f \times k \times X_a,$$

$$Y_{T_s} = f \times k \times Y_a, \quad ,$$

$$Z_{T_s} = f \times k \times Z_a.$$

where f*k* $X_aY_aZ_a$ means the estimated illuminant level [AMD1-4].

6) If $X_{T_s}, Y_{T_s}, Z_{T_s}$ have the same values with the previous values, go to procedure 7), else remove pixels that have the pixel value bigger than the self-luminous threshold and repeat procedure 4) to 6)

If ($X_{T_s}(t) = X_{T_s}(t-1), Y_{T_s}(t) = Y_{T_s}(t-1), Z_{T_s}(t) = Z_{T_s}(t-1)$) { go to 7) }

else {

$$\begin{cases} X(i,j) > X_{T_s} \text{ or } Y(i,j) > Y_{T_s} \text{ or } Z(i,j) > Z_{T_s}, & p(i,j) = 0 \\ otherwise, & p(i,j) = 255 \end{cases}$$

}

repeat 4) ~ 6)

where *t* means the iteration time for the T_s and the initial values are set to

$$X_{T_s}(0) = 0, Y_{T_s}(0) = 0, Z_{T_s}(t) = 0.$$

7) Averaging the *XYZ* value for all pixels remained, estimating it as an illuminant tri-stimulus value, and computing the scene-illuminant chromaticity coordinates (x_s, y_s) in CIE 1931 diagram. Again, *row * col* intuitively means the number of all pixels remained, which have *p(i, j) = 255.*

$$X_s = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} X(i,j),$$

$$Y_s = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} Y(i,j),$$

$$Z_s = \frac{1}{(row \times col)} \sum_{i=0}^{row-1} \sum_{j=0}^{col-1} Z(i,j).$$

$$x_s = \frac{X_s}{X_s + Y_s + Z_s},$$

$$y_s = \frac{Y_s}{X_s + Y_s + Z_s}.$$

8) Converting the scene-illuminant chromaticity (x_s, y_s) into color temperature T_c.

(1) Calculating the chromaticity coordinates (u_s, v_s) in CIE 1960 UCS diagram from (x_s, y_s).

$$u_s = \frac{4x_s}{-2x_s + 12y_s + 3},$$

$$v_s = \frac{6y_s}{-2x_s + 12y_s + 3}.$$

(2) Finding two adjacent isotemperature lines [Mori et al (1968)] from (u_s, v_s) and obtaining the distance from those lines: if (u_s, v_s) is located between i-th and i+1-th isotemperature line then di / di+1 < 0

$$d_i = \frac{(v_s - v_i) - t_i(u_s - u_i)}{(1 + t_i^2)^{1/2}},$$

where (u_i, v_i), t_i: chromaticity coordinates and slope for representing the i-th isotemperature line (Table AMD1-1 - Isotemperature lines: Calculated in accordance with the method proposed by Mori et al.(1968): The color temperatures between 1667K and 25000K and corresponding parameters(u_i, v_i, t_i) are marked with blue fonts) and d_i : distance between (u_s, v_s) and the ith isotemperature line.

(3) Calculating the correlated color temperature using the ratio of distance

$$T_c = \left[\frac{1}{T_i} + \frac{d_i}{d_i - d_{i+1}} \left(\frac{1}{T_{i+1}} - \frac{1}{T_i} \right) \right]^{-1},$$

where T_I is the color temperature for the cross point of the i-th isotemperature line with the daylight locus. The color temperatures less than 1667K and larger than 25000K are tuned to 1667K and 25000K, respectively.

Table AMD1-1 — Isotemperature lines: Calculated in accordance with the method proposed by Mori et al.(1968)

i	Reciprocal Megakelvin	TemperatureT (K)	u_i	v_i	t_i
1	0	Infinity	0.18006	0.26352	-0.24341
2	10	100,000	0.18066	0.26589	-0.25479
3	20	50,000	0.18133	0.26846	-0.26876
4	30	33,333	0.18208	0.27119	-0.28539
5	40	25,000	0.18293	0.27407	-0.30470
6	50	20,000	0.18388	0.27709	-0.32675
7	60	16,667	0.18494	0.28021	-0.35156
8	70	14,286	0.18611	0.28342	-0.37915
9	80	12,500	0.18740	0.28668	-0.40955
10	90	11,111	0.18880	0.28997	-0.44278
11	100	10,000	0.19032	0.29326	-0.47888
12	125	8,000	0.19462	0.30141	-0.58204
13	150	6,667	0.19962	0.30921	-0.70471
14	175	5,714	0.20525	0.31647	-0.84901
15	200	5,000	0.21142	0.32312	-1.0182
16	225	4,444	0.21807	0.32909	-1.2168
17	250	4,000	0.22511	0.33439	-1.4512
18	275	3,636	0.23247	0.33904	-1.7298
19	300	3,333	0.24010	0.34308	-2.0637
20	325	3,077	0.24702	0.34655	-2.4681
21	350	2,857	0.25591	0.34951	-2.9641
22	375	2,677	0.26400	0.35200	-3.5814
23	400	2,500	0.27218	0.35407	-4.3633
24	425	2,353	0.28039	0.35577	-5.3762
25	450	2,222	0.28863	0.35714	-6.7262
26	475	2,105	0.29685	0.35823	-8.5955
27	500	2,000	0.30505	0.35907	-11.324
28	525	1,905	0.31320	0.35968	-15.628
29	550	1,818	0.32129	0.36011	-23.325
30	575	1,739	0.32931	0.36038	-40.770
31	600	1,667	0.33724	0.36051	-116.45

6.9.1.1.3 Optimal Interval Determination for Color Temperature Browsing Categories

To find the optimal range of color temperature values for each browsing category, interval classifiers for fuzzy categories based on rough information systems were used.

6.9.1.2 Browsing Method

1. For the hot image browsing, the browser starts displaying the images from the lowest sub-range in the hot color temperature range and continues displaying the images in the subsequent sub-ranges.

2. For the warm and moderate image browsing, the browser starts displaying images in the middle sub-range and continues displaying the images in the sub-ranges near the middle sub-ranges.

For the cool image browsing, the browser starts displaying the images from the highest sub-range in the cool color temperature range and continues displaying the images in the subsequent sub-ranges in a descending order.

The following is a pseudo-code in HTML format for color temperature browsing in the web browser using DOM and JavaScript. This code reads the XML document and generates corresponding DOM objects. Assume that the XML document of image DB is composed of image elements, which are again composed of an image link and its color temperature browsing type. This code produces category buttons on the web window. If one of the 4 category buttons is pushed, it will return the images belonging to the chosen category. Here, SortAscendingOrder(), SortDecendingOrder(), and RearrangeNeartoFar() functions are left out to implementers where one can easily implement them.


```
<html>
    <head>
        <title>color temperature browsing</title>
    </head>

    <script language=”JavaScript”><!--Java script-->
    function FindHot(){
        dom = ColorTemperature.XMLDocument;<!--DOM Object declaration-->
        node = dom.getElementsByTagName(“Image”) ;
        resulth = “”;
        result.innerHTML = “”;
        node = SortAscendingOrder(node); <!-- pseudo code: sort nodes
                                             in ascending order by its
                                             subRangeIndex value -->
         for (i=0;i< node.length-1; i++) {
            TitleString=node.item(i).lastChild.firstChild.firstChild.nodeValue;
            if (TitleString == “hot”) {
                resulth=node.item(i).firstChild.firstChild.nodeValue+“\n\n”;
                result.innerHTML += “<img src=” + resulth + “> <br />”;
            }
        }
        if (resulth == “”)
            result.innerHTML = “&ltno hot image found&gt”;
    }

    function FindWarm(){
        dom = ColorTemperature.XMLDocument;<!--DOM Object declaration-->
        node = dom.getElementsByTagName(“Image”) ;
        node=RearrangeNearToFar(node); <!-- pseudo code: rearrange nodes
                                          with a middle value by
                                          its SubRangeIndex value -->
        resulth = “”;
        result.innerHTML = “”;
        for (i=0;i< node.length-1;i++) {
            TitleString=node.item(i).lastChild.firstChild.firstChild.nodeValue;
```



```
            if (TitleString == “warm”) {
                resulth = node.item(i).firstChild.firstChild.nodeValue+ “\n\n”;
                result.innerHTML += “<img src=” + resulth + “> <br />”;
            }
        }
        if (resulth == “”)
                result.innerHTML = “&ltno warm image found&gt”;
    }

    function FindModerate(){
        dom = ColorTemperature.XMLDocument;<!--DOM Object declaration-->
        node = dom.getElementsByTagName(“Image”) ;
        node = RearrangeNearToFar(node); <!-- pseudo code: rearrange nodes
                                               with a middle value by its
                                               SubRangeIndex value -->
        resulth = “”;
        result.innerHTML = “”;
        for (i=0;i< node.length-1;i++) {
            TitleString=node.item(i).lastChild.firstChild.firstChild.nodeValue;
            if (TitleString == “moderate”) {
                resulth=node.item(i).firstChild.firstChild.nodeValue + “\n\n”;
                result.innerHTML += “<img src=” + resulth + “> <br />”;
            }
        }
        if (resulth == “”)
            result.innerHTML = “&ltno moderate image found&gt”;
    }

    function FindCool(){
        dom = ColorTemperature.XMLDocument;<!--DOM Object declaration-->
        node = dom.getElementsByTagName(“Image”) ;
        node = SortDescendingOrder(node); <!-- pseudo code: sort nodes in
                                              Descending order by its
                                                 SubRangeIndex value -->
        resulth = “”;
        result.innerHTML = “”;
        for (i=0;i< node.length-1;i++) {
            TitleString=node.item(i).lastChild.firstChild.firstChild.nodeValue;
            if (TitleString == “cool”) {
                resulth=node.item(i).firstChild.firstChild.nodeValue + “\n\n”;
                result.innerHTML += “<img src=” + resulth + “> <br />”;
            }
        }
        if (resulth == “”)
            result.innerHTML = “&ltno cool image found&gt”;
    }
    </script>

    <body>
        <xml id=”ColorTemperature” src=”CTBrowsing.xml”></xml>
        <h1>Color Temperature Browsing</h1><p>
        <BUTTON ONCLICK=’FindHot()’>HOT</BUTTON>   
        <BUTTON ONCLICK=’FindWarm()’>WARM</BUTTON>   
        <BUTTON ONCLICK=’FindModerate()’>MODERATE</BUTTON>   
        <BUTTON ONCLICK=’FindCool()’>COOL</BUTTON>
        <hr>
        <div id=”result”></div>
        </body>
</html>
```