

Representation and Retrieval of Color Image Using Binary Space Partitioning Tree

G. Qiu and S. Sudirman
Computer Vision Group, School of Computer Studies
University of Leeds, Leeds, United Kingdom

Abstract

A new technique has been developed for the construction of binary space partitioning tree for color image representation. Based on the binary quaternion moment preserving thresholding, a color image is first binarised, and a goodness of fit criterion is then introduced to determine the partitioning line. An intuitive, yet simple, algorithm has been introduced to compare the contents of color images by matching their BSP trees. Experimental results are presented to demonstrate the performance of the method.

Introduction

Image representation has been a popular research area for the past few years. Recent approaches to represent images can be found in [8],[11]. Although it is evident that one of the driving forces in image representation research is to efficiently coding the image, it is not suggested until recently[7] that this area has a significant potential for content access application in the representation domain.

Effective and efficient representation of image data is essential in content based image indexing and retrieval. Existing methods in this ever-popular research and application area use color, texture and shape features as indexing and retrieval cues.[2,4,5,9,13] The emphasis was on extracting these features and using them either individually or in combination effectively. However, current state of the art computer vision/image processing techniques are still not mature enough yet to ensure accurate segmentation of the images into meaningful regions. Even if accurate segmentation can be performed, there is the added problem of representing the images into constituent regions effective and efficiently for indexing and retrieval purposes.

In this paper, we seek an intermediate solution. We would like to represent images with efficient data structures, which can be effectively and conveniently used for image content description. We do not seek highly accurate representation of the image in every detail; instead, the goal is to represent the image contents well enough to enable effective and efficient indexing and retrieval. We use binary space partitioning (BSP) tree[1] for image data representation. Although researchers in the past have used BSP tree for image representation,[8] it was only applied to gray scale images and extension to color image is not straightforward.

We have developed a novel method to build BSP tree for color image. Instead of using the modified Hough-transform or by minimizing the least square error to find the partitioning line,[8] we propose to apply the moment-preserving thresholding technique[6,15] to the color image and determine the line from the threshold image. As an application, we also introduce a scheme for comparing the similarity of color images by matching their BSP tree representations.

The rest of the paper is organized as follows. Section 2 describes the new technique for building the BSP tree for color image representations. Section 3 introduces a method for content-based comparison of color images by matching their BSP tree representation. Section 4 presents the experimental results and a concluding remark is given in section 5.

BSP Tree Representation of Color Images

The purpose of partitioning an image is to divide the image into a number of polygons containing data of similar characteristics. One simple method is to partition the image according to the raw data values, i.e., for a gray-level image the polygons should contain a part of the image having a similar gray-level. The extension of the idea to color image is not straightforward. Should the algorithm for a gray-level image be applied independently to each of the three channels of color data or should we work directly on a 3-D color space? The authors believe that treating the input data as a point in a 3-D space gives a more meaningful result than just considering the three components independently. This is the underlying reason for using the binary quaternion moment preserving thresholding method. The original idea of moment-preserving thresholding technique[15] was designed for grey-level image and is briefly reviewed here. Given a grey-scale image f with N number of pixels and its grey-level value at pixel location (x,y) denoted by $f(x,y)$, the i^{th} grey-moment is calculated as:

$$m_i = \Sigma_x \Sigma_y f^i(x,y) / N \qquad i=0,1,2,\ldots \qquad (1)$$

Let $F(f) \rightarrow g$ be a function that operates on f such that the 1^{st} to the i^{th} moment of g are equal to the 1^{st} to the i^{th} moment of f respectively. Then F is said to preserve up to the i^{th} moment of the input data in the output data. Arguably the more moments the operator preserves the more information of the input image is retained in the output image.[16] The output image g has only two grey-level values h_0 and h_1 which correspond to the pixels which values are below and above a threshold T respectively. In order to preserve the moment of the input image in the output image the following constraints have to be met:

$$p_0 + p_1 = 1 \tag{2}$$

$$p_0 h_0^{\,k} + p_1 h_1^{\,k} = m_k \qquad k \le i \tag{3}$$

where i is the maximum order of moment to be preserved and p_0 and p_1 denote the fraction of pixels having grey values h_0 and h_1 respectively.

Pei and Cheng[6] extended the idea to color image using quaternion numbers. The algebra of quaternion is the generalization of complex numbers.[3] The three-dimensional data set *(R,G,B)* of color images can be represented as a quaternion numbers *q(n)*:

$$q(n) = q0(n) + q1(n).i + q2(n).j + q3(n).k \tag{4}$$

where

$$q0(n) = 0;\ q1(n) = R;\ q2(n) = G;\ q3(n) = B$$

The three quaternion moments are defined as follows:

$$m_1 = E[q];\ m_2 = E[q.q^*];\ m_3 = E[q.q^*.q] \tag{5}$$

where q^* and $E[q]$ are the complex conjugate and the expected value of q respectively.

Instead of two grey values two quaternion numbers z_0 and z_1 are used.

$$z_0(n) = z_{00}(n) + z_{01}(n).i + z_{02}(n).j + z_{03}(n).k \tag{6}$$

$$z_1(n) = z_{10}(n) + z_{11}(n).i + z_{12}(n).j + z_{13}(n).k \tag{7}$$

In addition, the scalar threshold term T is extended to a 3-D hyperplane l_T. defined by:

$$a.q0 + b.q1 + c.q2 + d.q3 + e = 0 \tag{8}$$

which solution is given in [6] as:

$$a = -1;\ b = -(z_{01}-z_{11}) / (z_{00}-z_{10}) \tag{9}$$

$$c = -(z_{02}-z_{12}) / (z_{00}-z_{10});\ d = -(z_{03}-z_{13}) / (z_{00}-z_{10}) \tag{10}$$

$$e=(z_{00}^{\,2}+z_{01}^{\,2}+z_{02}^{\,2}+z_{03}^{\,2}-z_{10}^{\,2}-z_{11}^{\,2}-z_{12}^{\,2}-z_{13}^{\,2}) /(2.(z_{00}-z_{10})) \tag{11}$$

$$z_{03} = -(a_2-\sqrt{(a_2^{\,2}-4a_1c_{00})}) / 2a_1 \tag{12}$$

$$z_{13} = -(a_2+\sqrt{(a_2^{\,2}-4a_1c_{00})}) / 2a_1 \tag{13}$$

$$z_{l0} = (w.u+v).z_{l3};\ z_{l1} = (w.v-u).z_{l3};\ z_{l2} = w.z_{l3} \quad l = 0,1 \tag{14}$$

$$u = (c_{10}.c_{12} - c_{11}.c_{13})/(c_{13}^{\,2}+c_{12}^{\,2}) \tag{15}$$

$$v = (c_{11}.c_{12} - c_{10}.c_{13})/(c_{13}^{\,2}+c_{12}^{\,2}) \tag{16}$$

$$w = (c_{11}.v+c_{10}.u+c_{12})/(c_{10}.v-c_{11}.u+c_{13}) \tag{17}$$

$$a_1 = (1+w^2)(1+u^2+v^2) \tag{18}$$

$$a_2 = w.(u.c_{10}+v.c_{11}+c_{12})-u.c_{11}+v.c_{10}+c_{13} \tag{19}$$

The image is first binarized using the binary quaternion moment-preserving thresholding method. A partitioning line is then chosen to divide the output image into two regions such that at least one of the regions is relatively homogenous, i.e., for a binary image it is either almost or completely black or white. However, we do not require the partition line to always produce completely homogenous regions since the white and black pixels in the binary form of a natural image will almost be scattered everywhere. Imposing this constraint tends to produce very small and less meaningful region. As a result, the ability to control the homogeneity of the resulted partitioned region is required as different applications may require different homogeneity criteria. For this purpose, the parameters of the partitioning line, θ and ρ of a binary image *g(x,y)* are chosen as:

$$(\theta,\rho) =$$

$$arg\ Max_{\forall\theta,\rho}\ \{Max[(S_l^{z0})^{\kappa}/N_l, (S_l^{z1})^{\kappa}/N_l, (S_r^{z0})^{\kappa}/N_r, (S_r^{z1})^{\kappa}/N_r]\} \tag{20}$$

where

$$S_l^{zi} = \{\Sigma\, n^{zi}(x,y) \mid x.\cos(\theta)+y.\sin(\theta) < \rho\}$$

$$S_r^{zi} = \{\Sigma\, n^{zi}(x,y) \mid x.\cos(\theta)+y.\sin(\theta) \ge \rho\}$$

$$n^{zi}(x,y) = 1 \text{ if } g(x,y) = z_1 \text{ or } 0 \text{ otherwise. } i = 1,2$$

$$N_l = S_l^{z0} + S_l^{z1};\ N_r = S_r^{z0} + S_r^{z1};\ \kappa = \{\Re \mid \kappa > 1\}$$

The value of κ determines the allowable impurities in the partitioned region scaled by the size of the region. For example, we have two possible lines, l_1 and l_2. The first line produces a region of size 100 pixels and all of the pixels have value z_0. The second line produces a region of size 1000 pixels but only 800 of which have value z_0. If κ=1.1 is used then the first line is chosen and if κ=1.2 the second line is chosen instead.

To obtain the best partitioning line the function described in Equation 20 is calculated for every possible line passing through the image. In practice, since the number of lines is infinite then quantization of the line is necessary. This is achieved by quantizing the line parameters θ and ρ.

The range of possible values of θ is a periodic function of period π. Therefore, choosing the range $-\pi/2 \le \theta < \pi/2$ is sufficient to represent all possible values of θ. In this paper θ is quantized into 16 equally spaced values. The choice is arbitrary and is affected by the amount of computational cost and the required accuracy of the partitioning line.

The quantization of ρ however is not that trivial since it depends on θ. The formulation for possible range and the best quantization step of ρ is given in [8]. The set of lines with orientation θ that passes through an image has range value $[\rho_{min}, \rho_{max}]$ where:

$$\rho_{min}(\theta)=min\ \{x.cos\theta+y.sin\theta\} \quad (21)$$

$$\rho_{max}(\theta)=max\ \{x.cos\theta+y.sin\theta\} \quad (22)$$

where (x,y) is the set of coordinates of the image vertices. The quantization step $\partial\rho$ is calculated by:

$$\partial\rho(\theta) = max\{|\ cos\theta\ |\ ,|\ sin\theta\ |\} \quad (23)$$

Figure 1. Images partitioned using the proposed method with κ=1.35. From left to right and top to bottom order, Original image, 1st, 2nd and 3rd partition levels (edge enhanced)

After obtaining the best partitioning line, one color is chosen to represent the part of the input image contained in each region. In the interest of computational speed, this paper calculates the element values of the representative color as the mean of the red, green and blue components of all the pixel colors in the region. These color values together with the partitioning line parameters are recorded and they are used as the representation of the image at the first partition level.

The process, starting from 1) thresholding, 2) finding an optimal line to, 3) the calculation of the mean color, is repeated for each region. The data used for the thresholding is the data from the original image. However, this time instead of thresholding the whole image, we threshold the regions resulted from the previous process independently of each other. A region will not be partitioned if it becomes too homogenous or if it becomes too small. The homogeneity criterion used is the value of the second moment m_2. If it becomes too small than the partitioning for that region is stopped. The process is repeated until no more regions can be partitioned or until it reached certain number of iteration. Therefore, at the end of the j^{th} iteration one has j number of image representations at a hierarchical order. Figure. 1 shows the partitioning of natural images using the proposed method

Color Image Matching based on BSP Tree

The correspondence between the image regions, the partitioning lines parameters and the nodes of the BSP tree is illustrated in Figure. 2. The root node contains the average color of the whole image and the first partitioning line l_1. The leaves contain the average colors of the regions in the subdivision that the BSP induces and the parameters of the lines that partitioning the regions. For example in Figure.2, node b contains the average color of region b and the line l_2 that partitioning region b. With this BSP tree representation, we introduce a method for comparing image contents.

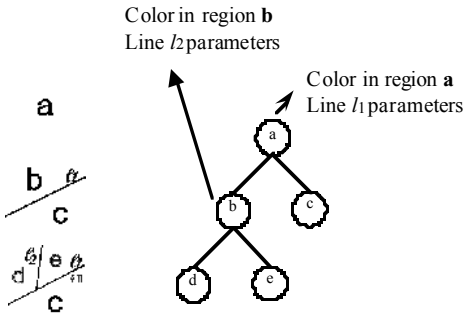

Figure 2. a BSP-tree representation of an image.

In our view, the process of matching two trees is one that compares the properties of the nodes in one tree to those in the other, and find the minimum difference between them while taking into account the hierarchical structure of the trees. To do this first we define an entity called *node-family* of a tree u, denoted by $\boldsymbol{N}_u$. A node family $\boldsymbol{N}_u(i,j)$ is a sub-tree of u consisting one node of u at the i^{th} level and its two sibling nodes (see Figure 3) and j is the index.

Let $\boldsymbol{N}_k(i,j)$ denote the j^{th} node family at layer i of tree k, and $\boldsymbol{N}_l(m,n)$ denote the n^{th} node family at layer m of tree l. Comparison of the contents of the two images represented by trees k and l is performed as follows: For each $\boldsymbol{N}_k(i,j)$, for all i and j, calculate the node family differences $\boldsymbol{Nd}$ (definition given later) between $\boldsymbol{N}_k(i,j)$ and all the node families of the tree l at levels i-1, i, and i+1, and take the minimum node family difference value and denote it as $\boldsymbol{D}_{i,j}$. The difference between the trees $\boldsymbol{DT}_{k\text{-}l}$ is the average of $\boldsymbol{D}_{i,j}$ over all i and j. The same process is repeated for $\boldsymbol{DT}_{l\text{-}k}$ since $\boldsymbol{DT}_{k\text{-}l}$ is not necessarily equal to $\boldsymbol{DT}_{l\text{-}k}$. And the overall tree difference $\boldsymbol{DT}(k,l)$ is calculated as the average of the two,

$$\boldsymbol{DT}(k,l) = (\boldsymbol{DT}_{k\text{-}l} + \boldsymbol{DT}_{l\text{-}k})/2 \quad (24)$$

Let j and n be two node families. Let L_{jp} and L_{np} be the lines associated with the parent nodes of j and n respectively. Let C_{jl} and C_{jr} be the colors associated with the left and right child node of j respectively and let C_{nl} and C_{nr} be the colors associated with the left and right child node of n respectively. The node family difference between j and n, $\boldsymbol{Nd}(j,n)$ is defined as the sum of the differences between the lines and the colors, i.e.

$$Nd(j,n)=D(L_{jp}, L_{np})+(D(C_{jp},C_{nl}) + D(C_{jr}, C_{nr}))/2 \quad (25)$$

where

$$D(L_{jp}, L_{np}) =(D(\theta_{jn})+D(\rho_{jn}))/2$$

$$D(\theta_{jn}) = min(|\theta_j-\theta_n|, \pi-|\theta_j-\theta_n|)$$

$$D(\rho_{jn}) = |\rho_j-\rho_n|$$

$$D(C_{jS},C_{nS}) = \sqrt{\{(R_{jS}-R_{nS})^2+(G_{jS}-G_{nS})^2+(B_{jS}-B_{nS})^2\}} \qquad S=l,r$$

Because the line parameters and colors represent difference modalities, care must be taken in combining them together. One difficulty in combining the distances of the partitioning lines and the distance of colors is that they represent a priori not comparable modalities, with different dynamic ranges. In the absence of any systematic method for combining different modalities, we propose to normalize all difference entities before combining them.

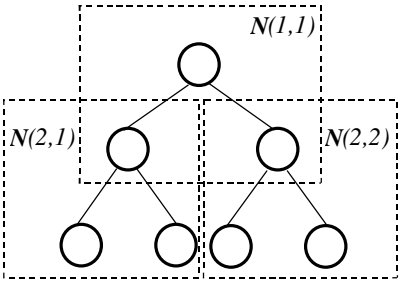

Figure 3. The BSP nodes are divided into node family units, as shown inside the dash line boxes, which contains the parent and its two children

In our co-ordinate definition, the origin is the top-left hand corner of the image, the values of θ range from $-\pi/2$ to $\pi/2$. If we denote the image height and width by H and W respectively, the values of ρ therefore range from $-H$ to $\sqrt{(H^2+W^2)}$. The RGB values of the color has a range between 0 – 255. Before calculating the distances, θ's are normalized to [-0.5,+0.5], ρ's are normalized to (-0.5,+0.5] and RGB's are normalized to [0,1]. In order to reduce the time taken to calculate the tree difference, $\boldsymbol{Nd}$ is calculated only for all node families up to an arbitrary level instead of for all node families in the trees.

Color Image Matching Experiment

The image comparison method was implemented in a database consists of 1065 images using 61 pairs of test images, some of which are shown in Figure 6. The test image set is a subset of the database image set. They were manually handpicked such that each pair consists of similar scenes or similar objects. The value of κ in Equation 20 is set to 1.35. The tree matching were done up to partition level 5. For each query image, the differences between its tree and that of the images in the database are calculated and the result is sorted in ascending order. To measure the matching performance, we calculate the matching percentile defined as

$$MP = 100 \times (N\text{-}R) / (N\text{-}1) \quad \text{if } R \le 10 \text{ or } 0 \text{ otherwise} \quad (26)$$

where N is the total number of images, R is the rank of the corresponding pair of the query image.

As a comparison, we also implemented the color correlogram method.[4] The experiment showed a very promising result. The average matching percentile for all 61 pairs of test images using the BSP method is 94.2 and that of color correlogram is 93.4. We believe the success of our new method can be attributed to the fact that it not only utilizes the color but also the structure information of the image as well. Figure 4 shows some of the test image pairs from which BSP method outperforms color correlogram.

The computation time required to partition an image depends on the quantization step of the line parameters and the size of the image. The computation time required to match the tree linearly depends on the size of the database and the number of nodes compared in each tree. In practice, the BSP tree of the images in the database is generated and stored before any matching is done. Therefore, during matching process the computational load lays only on the partitioning of the query image and the matching of the tree.

Figure. 4. Three sample pairs of test images in which color correlogram fails whilst our method succeeds.

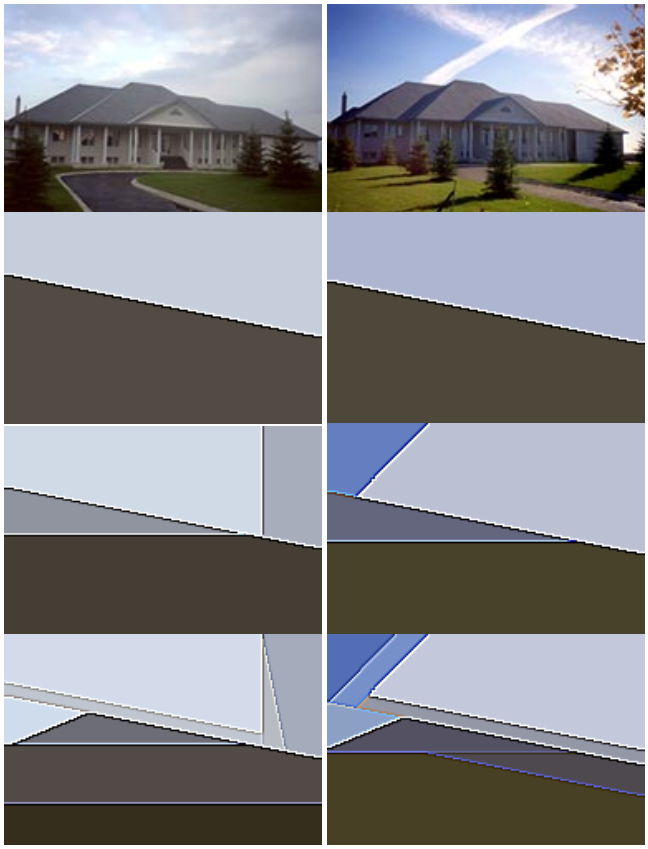

Figure 5. The partitioning of two similar image both of which contain the same object but taken from slightly different viewpoints and under different weather conditions. Top to bottom order, original image, 1st, 2nd, 3rd partition levels (edge enhanced)

Concluding Remarks

A new method to represent color image contents using the BSP tree has been developed. An intuitive algorithm for matching color image contents through the matching of their representation trees has also been introduced. We have presented some experimental results, which demonstrated good performance and clear potential. The technique is still in an early stage of its development. Our current work continues the study of how to partition the image effectively and efficiently, and the development of new matching methods for comparing image contents based on the BSP tree representation.

The method has wider applications also. The obvious one is image coding.[8] The advantage of this approach is that the coded information can be used conveniently in content based indexing and retrieval.[12] Our ultimate goal is the development of the new generation of color image coding methods that meet that 4th criterion.[7]

References

1. M. de Berg, et al, Computational Geometry: Algorithms and Applications, Springer, 1997, pg.251
2. C. Carson., et.al., "Region-based image querying", Content-Based Access of Image and Video Libraries, Proceedings, pg. 42-49, (1997)
3. J.B. Fraleigh, A First Course in Abstract Algebra. Reading, MA: Addison-Wesley, 1982
4. J. Huang, et.al., "Image indexing using color correlogram", Computer Vision and Pattern Recognition, Proceedings., pg. 762-768, (1997)
5. W.Y. Ma and B.S. Manjunath, "NeTra: A toolbox for navigating large image databases", Multimedia Systems, vol. 7, pg. 184 –198, (1999)
6. S.C. Pei and C.M. Cheng, "Color image processing by using binary quaternion moment-preserving thresholding", *IEEE* Trans. on Image Processing, vol. 8, pg.614-628, (1999)
7. R.W. Picard, "Content Access for Image/Video Coding: "The Fourth Criterion"", MIT Media Lab Perceptual Computing Section Technical Report, No. 295, Statement for Panel on "Computer Vision and Image/Video Compression", ICPR94, Jerusalem, (1994)
8. H. Radha, M. Vetterli, R. Leonardi, "Image compression using binary space partitioning tree", IEEE Trans. on Image Processing, vol.5, pg.1610-1624, (1996)
9. Y. Rui, et.al., "Image retrieval: Current techniques, promising directions, and open issues", J. of Visual Communications and Image Representation, vol. 10, pg. 39 –62, (1999)
10. P.K. Sahoo, S. Soltani and A.K.C. Wong, "A survey of thresholding techniques", Computer Vision Graphic and Image Processing, pg. 233-260, (1988)
11. P. Salembier, L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval", IEEE Trans. Image Processing, vol. 9, pg. 561-576, (2000)

12. G. Schaefer and G. Qiu, “Midstream content access based on visual pattern coding”, SPIE Proc. Storage and Retrieval for Media Database 2000, vol. 3972, pp.284 – 292, (2000)
13. M.J. Swain, and D.H. Ballard, “Color Indexing”, International Journal of Computer Vision, Vol. 7, no. 1, pp. 11-32, (1991)
14. A.J. Tabatabai and O.R. Mitchell, "Edge location to sub-pixel values in digital imagery", IEEE. Trans. Pattern Analysis and Machine Intelligence, vol.6, pg. 188-201, (1984)
15. W.H. Tsai, "Moment preserving thresholding: A new approach", Computer Vision Graphic and Image Processing, pg. 377-393, (1985)
16. C.K. Yang and W.H. Tsai, "Reduction of color space dimensionality by moment-preserving thresholding and its application for edge detection in color images", Pattern Recognition Letters, vol.17, pg.481-490, (1996)

Biography

Guoping Qiu received his BSc. degree in Radio Engineering from University of Electronic Science and Technology of China and received his PhD in the areas of neural network and image processing from University of Central Lancashire, England. He was a lecturer in University of Derby and now in University of Leeds, England. He is a member of IEEE and his research interests include Neural Network, Image Database and Image Coding, Web Informatics and Industrial Inspection.

Sudirman received his BEng. degree in Control System Engineering, University of Sheffield, England He is currently a PhD candidate at University of Leeds with research area in image representation for coding and indexing.

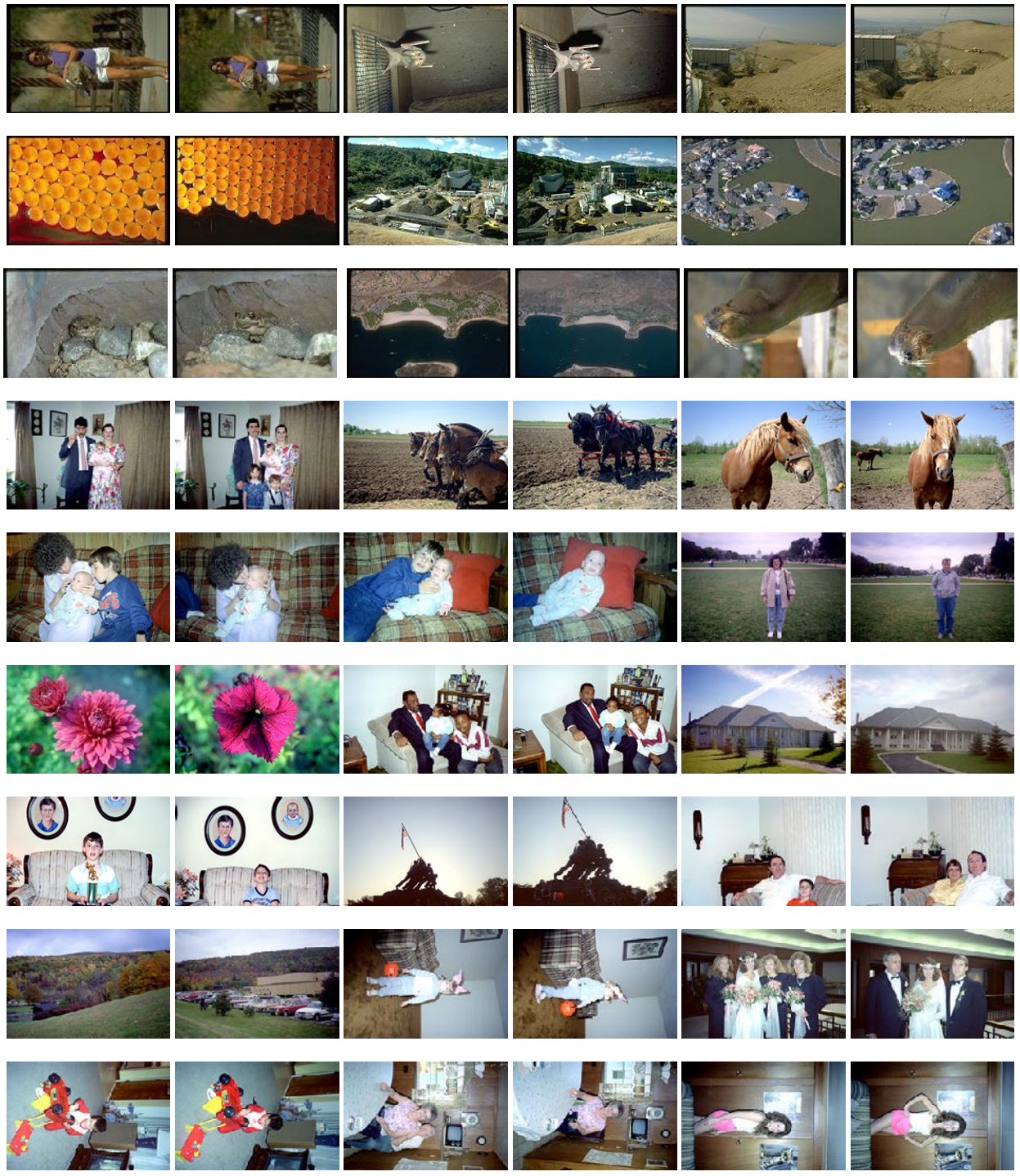

Figure 6. Thirty pairs out of 61 pairs of test images used in the experiment