Abstract

In the framework of moving image analysis for image coding, the time coherence of segmentation is important for at least two reasons. In the first place, when the user is allowed to interact with the encoded scene, it must be possible, for instance, for him to select, zoom, rotate, or change any scene objects. This implies that objects must be identified uniquely in each of the images of the sequence in which they occur. Finally, segmentation coherence is important because it improves coding efficiency when time prediction is used.

This paper extends previous work on still image segmentation using split & merge [1] to deal also with moving images. The technique proposed performs a time-recursive segmentation. Time recursion is introduced by using a region-growing like version of the original segmentation algorithm such that the previously segmented regions are projected into the current image. The results obtained show that the technique is a good candidate for image analysis in the framework of second-generation moving image coding.

Introduction

Pixel-based moving image coding techniques are well established. The last international standards using these so-called first generation techniques have been issued in the past few years (e.g., H.261, MPEG-1, MPEG-2 and H.263). Second-generation techniques, which attempt to address moving images using middle level vision concepts, have been maturing during the last decade and new standards are being developed.

A significant amount of today’s research in moving image coding using second-generation techniques is being done in the framework of MPEG-4 [2], which aims at providing: (a) easy access and manipulation of the contents of image sequences—in terms of scene objects—, apart from providing the usual requirements of compression, quality, and cost, and (b) a flexible syntax, which will enable the standard to “survive” longer by allowing and encouraging evolution. MPEG-4 will not, however, standardise the means of automatically obtaining a scene description, i.e., it will not specify how to analyse moving images. Without such an analysis, an MPEG-4 compliant coder will be essentially identical to the classical pixel-based coder (with added syntax capabilities).

One of the most important tasks of moving image analysis is segmentation, i.e., the process of decomposing a sequence of images into a set of objects that are meaningful according to some criteria. Segmentation should be coherent in time in order to facilitate user interaction and to improve coding efficiency. This paper proposes an extension to the third dimension (time) of the two-dimensional segmentation techniques introduced in [3] and [1]. The extension is shown to be a good candidate towards achieving the aimed time coherence of segmentation.

1 2D segmentation algorithms

The definition of segmentation is apparently simple: produce a partition\(^1\) of the set of image pixels so that each set (region) in the partition is uniform according to a certain criterion.

The first problem with this definition is that, even if an appropriate uniformity criterion is given, it establishes no constraint as to the number or connectivity of the regions. Hence, another, possibly more useful, definition may be: produce a partition of the image into a minimum set of connected regions such that a certain global uniformity measure is above a given threshold.

Or: produce a partition of the image into a certain number of connected regions such that a certain global uniformity measure is maximised.

The choice of appropriate uniformity measures depends on the task at hand. If the segmentation aims at identifying the objects in the image automatically, e.g. if the image is to be manipulated or edited easily by a human, then the measure will have to be related to the semantics content of the represented scene. Developing such measures is a daunting task that implies knowing with detail all the levels of the human visual system. However, by using simpler measures, one may render the problem tractable and still hope the results to be of some use for human manipulation.

1.1 RSST segmentation

The algorithm presented in [3] is conceptually simple. A two dimensional image is first considered as a graph having as many nodes as there are pixels in the original 2D image. The graph has one arc between any

\(^1\)A partition of a set \(S\) is a set \(R\) of subsets of \(S\) such that the union of all the elements of \(R\) is \(S\) and, for all \(r \neq s \in R\), \(r \cap s = \emptyset\).
two nodes corresponding to adjacent pixels in the image (e.g., in the 4-neighbourhood sense if the image grid is rectangular). At each step of the algorithm, two regions are merged together, that is, two nodes of the graph are merged into a single node. The links are updated correspondingly, such that the existence of and adjacency between two regions in the image is always reflected by the existence of a single arc between the corresponding nodes.

The order in which the regions are merged together is crucial. In [3] the regions which are merged in each step are those which, after merging, produce the minimum increase in the root mean square colour difference (error), given a certain model for the colour of each region. The model suggested in [3] is simple: the colour of each region is approximated by a constant function. In such case, the optimum value of each region is simply the average of the colours of each pixel, provided that the difference between two colours is given by the Euclidean distance between the two colours in the colour space used.

Of course, the successive merges do not guarantee that the attained partition will be optimal in the sense defined above. One of the strengths of this method, however, is that it can be implemented efficiently. Careful implementations have a complexity of the order of $N \ln N$, where $N$ is the number of pixels in the image (though such implementations tend to be quite demanding in memory resources).

The method proposed in [1] also uses successive merge steps, though with some differences. The first is that the order of the merges is done in terms of the difference between the average region colours, instead of aiming at minimising a global error measure. The second is that, due to the tendency of the above criteria to preserve many small regions, the merges are done in three steps, the second of which aims at eliminating those small regions that are deemed irrelevant. The last difference is that the merges are not done from the original region graph having one node for each single pixel. The merges start from a region adjacency graph (RAG) obtained by splitting the image in a quad-tree fashion according to a certain uniformity criterion (e.g., dynamic range).

The split phase of the algorithm is useful from the implementation point of view, since it greatly reduces the memory requirements. Actually, if the quad-tree is built in a breath first manner, the allocated memory can be used as one of the stopping criteria of the split phase.

It can be seen that both methods construct a spanning tree of the original image graph (at pixel level, or using the leaves of the quad-tree as the initial regions of the graph). The spanning trees can be said to recursively minimise the sum of the link weights in the tree, and hence the algorithms are called Recursive Shortest Spanning Tree (RSST) algorithms.

1.2 RSST extension using markers

The RSST algorithms may be stopped using two criteria: when the global approximation error exceeds a certain threshold, or when a given number of regions has been attained. These algorithms provide no means for controlling the position of the resulting regions or for specifying markers around which the regions of interest should be obtained, as happens in region growing algorithms (such as watersheds [4]). However, they can be easily extended with such features.

Consider a label image with the same size as the original image. Let label zero denote unlabelled pixels and class $l$, with $l \neq 0$, be the set of all pixels with label $l$. The set of existing classes plus the set of unlabelled pixels is a partition of the image. The labellings may be restricted to those having connected classes.

Both algorithms can be extended such that an initial labelling is taken into account during segmentation. When choosing the next two adjacencies to merge one may simply say that regions with different labels should be merged last, if ever, or that regions with the same label should be merged first. Further, the merging rule may specify that, whenever an unlabelled region is being merged with a labelled region, the resulting region will inherit the label of the labelled one. On the other hand, if two regions are being merged which have different labels, the resulting region may inherit either the highest label or the label of the largest region. Notice that, in the case of the algorithm in [1], the quad-tree split phase must be performed such that the resulting regions do not have pixels with different labels.\(^2\)

Figure 1.2 shows the result of segmenting the image in Figure 1.2 with the set of (single pixel) markers represented by crosses in Figure 1.2. There are only six different labels: background, plant, sofa, hair, face and body. The markers on the hair all share the same label, the same thing happening with the markers on the face and the background. Figure 1.2 shows the result of segmenting the same image without markers and stopping the segmentation when six regions are obtained. Both cases use the algorithm in [3] after a split phase as in [1].

2 Extension to moving images

The extension of the described algorithms to moving images is simple: stack the successive images of a sequence into a 3D image, consider the (generally non-planar) RAG obtained using a 3D 6-neighbourhood (if the image sequence grid is rectangular), and leave the rest of the algorithms unchanged.

\(^2\)Though some of the pixels in a region may be unlabelled, if all the other pixels have a single label.
2.1 Requirements

When segmentation of long sequences of 2D images is the aim, simply segmenting the 3D images obtained by stacking a few 2D images at a time may cause problems. The first is that the aim is to obtain a sequence of 2D partitions. For instance, a perfectly acceptable 3D partition with connected regions may lead to some 2D partitions containing disconnected regions. Also, if an object has undergone a large movement from one image to the next, it will result in two objects in 3D segmentation. It should, however, result in a single object whose location in two different images does not overlap.

Another problem has to do with the amount of images that should be stacked before performing 3D segmentation. Clearly, the ideal would be to stack as many images as possible. However, this can easily result in both an overwhelming amount of data to process and unacceptable delays when segmentation is to be performed in real time, because segmentation can proceed only when all the images are available. Also, no matter how many images one stacks before 3D segmentation, there will always come a time when the next stack of images will have to be processed. The coherence between the previous and the next partitions will then be lost, unless other measures are taken.

Concluding, the algorithms should be able to track regions along time, should not demand too much computational power, and should introduce small delays for real-time applications.

2.2 Time recursivity

A solution for the problem of maintaining temporal coherence in segmentation is described in [4]. The idea is to perform 3D segmentation on stacks of images that overlap along time, and use partitions obtained in the past segmentations as markers for the present segmentation, thus introducing time recursivity. The minimum configuration providing temporal coherence consists of stacks of two images, maintaining a time overlap of a single image. Since the RSST segmentation algorithms tend to consume a large amount of memory, the use of pairs of images is amply justified by implementation considerations.

When the past partitions are “grown” into the present through 3D segmentation, a connected 3D region in the obtained partition may turn to be disconnected if restricted to a smaller time range. For instance, in the pairwise 3D segmentation scheme suggested above, the 2D partition corresponding to the present image in a just segmented pair may have disconnected regions. This problem is solved easily if, after the 3D segmentation involving all the images in the stack, the segmentation proceeds using only the present images. Before that, however, the regions which were split into more than one connected component will be “unlabelled”, with the exception of one of the connected components. Usually the largest connected component retains the label of the originating region in the past. This simple solution thus may create regions with new labels, which did not exist in the immediate past.

Using time recursivity, some regions may not “grow” into the present, and hence regions, and the corresponding labels, may disappear when no correspondence is found from the past to the present. Further, some regions in the present may not correspond to any region in the past, and hence new labels may also appear this way.

The problem with time recursivity using the described methods is that the number of regions tends to grow. This is caused because of illumination effects which often create new (artificial) regions, no matter how complex the region models are, and because regions seldom disappear. Hence, it is desirable to complement the segmentation steps explained above with a further step in which no label restrictions are imposed, and hence some differently labelled regions can be allowed to merge.

3 Results

Figure 2 shows the results of segmenting the first image of the Table tennis sequence. The RGB colour space was used, and the target root mean square colour difference (or error) was 22 (corresponding to a PSNR of 21.3
It can be seen that the technique is able to discriminate visually important features in the image. It can be seen also that a “false contour” was introduced in the background. This is due to the flat region model used, which is not able to model slowly varying region colours. The simple region model used also accounts for the division of the sleeve into regions of different shading.

![Figure 2: Segmentation of the first image of Table tennis.](image)

The time coherence of the segmentation algorithm proposed can be seen in Figure 3, which shows the time evolution of ten classes\(^5\) of the segmented sequence corresponding to the arm, hand, and racket of the player. The algorithm introduces new regions when the approximation error is not good enough, as can be seen in the lower part of the arm from the eighth image on.

![Figure 3: Segmentation of images 1, 3, 5, 7, and 9 of Table Tennis. Ten classes are shown in different gray levels: hand (three classes), racket (one and two classes), sweater cuff (one class), sleeve and shoulder (three and four classes).](image)

---

3 A low PSNR target was chosen in order to obtain a final number of regions which would be meaningful on printed paper.

4 Which is the case of the background, though invisible on the printed images.

5 But only nine labels, since label ten, used initially for the border of the racket, is later reused in the lower part of the arm.

### 4 Conclusions

This paper described an extension of the algorithms in [1] and [3] which deals with sequences of images while maintaining coherence of the obtained partitions from image to image. The results obtained are good and show that the method is interesting for use in second-generation moving image coders.

Future work will centre around three main directions. The first is the introduction of better region models (e.g. affine or polynomial), in order to avoid the false contours, and illumination models, to avoid the artificial separation of regions which semantically are one only (see the arm in Figure 3). The second will be the introduction of motion compensation so as to improve the “projection” of past partitions into the future. Finally, control of the complexity of the region borders will be introduced, in order to avoid the often ragged contours obtained when the region model used is insufficient to represent the colour of the regions.

### 5 Acknowledgements

The author would like to thank JNICT (PRAXIS programme) for its support to this work.

### References


