Video-rate Image Segmentation by means of Region Splitting and Merging

Kanur Aneja, Florence Laguzet, Lionel Lacassagne, Alain Merigot
Institute for Fundamental Electronics, University of Paris South
Orsay, France
kanur.aneja@gmail.com, florence.laguzet@u-psud.fr
lionel.lacassagne@u-psud.fr, alain.merigot@u-psud.fr

***Abstract*—This paper proposes a fast method for image segmentation. After an optimal split of the image into rectangular regions, this paper focuses on the fast merging of these regions. Since the computation time is very small, hence it is suitable for real time applications, while producing a good segmentation for tracking purposes.**

I. Introduction

Image segmentation aims to group pixel in an image into regions, based on their similarity in terms of grey level, color or texture. It is an important step in practical application of image analysis as it is frequently a preliminary pass for object localization, recognition or tracking, etc. While many sophisticated methods have been proposed, most are presently unable to perform this segmentation on the fly, as it is required in many time-critical applications like vehicle guidance, and so on.

We present in the paper a very fast segmentation scheme based on image splitting and merging. A classical *Split and Merge* segmentation method was proposed in [1]. It splits the image recursively in squares until all the squares are homogenous enough. It has some drawbacks inspite of the fact that it is very time effective. It is incapable of adapting itself to the image characteristics i.e. it produces large number of regions for boundaries other than the horizontal and vertical ones. Also these region boundaries are highly dependent even on simple tranformations like translation and scaling. Hence we choose instead the *Optimal Split* method[2]. Here if a given rectangle is not homogenous enough, we select an optimal split position, i.e. a position that will lead to the most homogeneous subregions. The computation of this position can be very efficiently done with the help of the 'scans' or 'prefix sum' that can dramatically accelarate the computation of global sums over square regions, in order to calculate, for instance, the average intensity and variance of these regions. A binary tree data structure is used to maintain the list of these homogenous regions obtained during the split phase. The main interest of this approach is that the splitting is adapted to the image charateristics which largely reduces the initial number of regions without any significant change in the computational cost. This paper presents a new method for merging the regions obtained from the optimal split method. A region is merged with its neighbours till it can merge no further so as to have the variance of all the regions below a given threshold.

II. Implementation

Our segmentation method proceeds in the following manner. Firstly, in the split phase, the initial split is performed in order to get the representation of the image as a set of homogenous regions. Each homogenous region is assigned a unique label. Then a region adjacency graph (RAG) is constructed corresponding to this region set so that each region gets references to its neighbours. In the merging phase, every region is merged with its surrounding regions to get larger regions which are still homogenous enough. This RAG is updated until no more regions can be merged further. Merge synoptic (Fig. 1) is the flow diagram representing the various steps of our method. Here the boundary and rgb images have been used to help in understanding the algorithm mechanism. The next section presents the region merging algorithm and all the above mentioned aspects will be described in detail in the following sections.

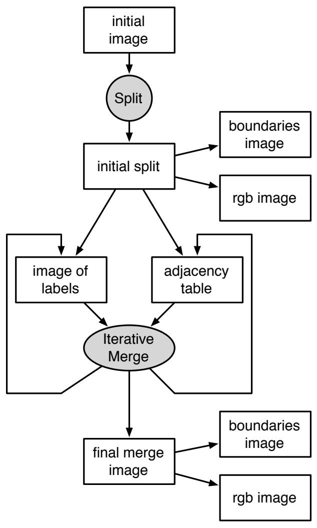


Fig. 1. Merge synoptic

A. Algorithm

B. Image of labels

During splitting, a binary tree data structure is used to maintain the list of the regions. So by scanning the binary tree, all the regions can be accessed. But only the leaf nodes of this tree will represent the homogenous regions obtained upon the termination of the splitting process. Each leaf node represents its corresponding homogenous region and hence while traversing the tree, the labels are assigned to these nodes and a new image of labels is then formed where each region is filled by its corresponding label.

Algorithm 1:

```
foreach valid region $e_k$ do
    select its best neighbour $f_k$
    if neighbour $f_k$ is valid then
        if ((gray level($e_k$) $\cong$ gray level($f_k$)) then
            Region is merged with its best neighbour if the
            homogeneity criterion is followed
            Update information for this new region in the
            following-:
              • Image of labels
              • Adjacency table (representing $RAG$ so formed)
              • Properties of the region after merging (average
                intensity, variance,size, etc)
            Continue merging the new region so formed
            (updated $e_k$) until it cannot merge any further
        else
            select another neighbour $f_k$ for $e_k$
        end
    else
        invalidate the region $e_k$ (remove it from the list)
    end
end
```

C. Adjacency table

This table is based on the image of labels so as to maintain the list of neighbours for each region in the original image after the split phase is over. Because of the optimal splitting process, the image is divided into several rectangular regions. So while scanning the image of labels, there are four possibilities which can be encountered by the scanning element (Fig. 2 top right)

- vertical border
- horizontal border between two adjacent regions having different labels
- corner formed by three adjoining regions and
- the case where the element remains inside the same region

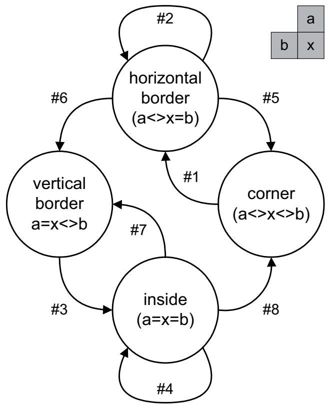


Fig. 2. Automaton and the scanning element

In order to accelerate the building of the adjacency table, a finite state automaton (Fig. 2) has been created to represent all the possible transitions. Hence, the labels are added to each other's list of neighbours if and only if, a vertical border or a corner is encountered by the scanning element (cases (5),(6),(7),(8) of Fig. 3). Because when the element moves inside (i.e. within) the same region, it does not encounter a transition. The transition in case of the horizontal border is insignificant because it ultimately leads to a vertical border or a corner. While adding the label to the neighbours list of a particular region (i.e. represented by another label) it is also ensured beforehand that that label is not present in the list being considered. As the merging progresses, this adjacency table is modified to contain the combined list of neighbours of the regions being merged.

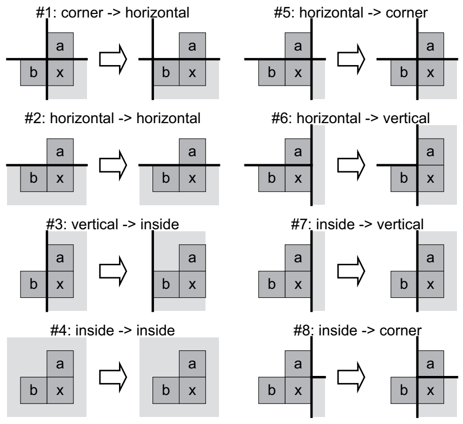


Fig. 3. Transitions

D. Selection of the next region to merge

It can be done using several methods:

- the region is selected randomly from the list of available valid regions
- the region having minimum variance(i.e. the most homogenous region) is selected every time from the updated list of valid regions
- the region with the largest size can be selected to commence the merging process

It appears that these methods lead to similar results in terms of the quality of the segmentation. As a simple random selection is much faster, this is what will be used in this paper.

E. Best neighbour selection

Every region in the image obtained after splitting has its own list of neighbours (i.e connected regions) as contained in the adjacency table. If the difference of average gray level intensities of the selected region and its neighbour is less than the average threshold (0.4 $\times$ variance threshold gives good results), then the neighbour is considered to be eligible. This is done to check the similarity of the grey levels of the regions being merged because otherwise, by only considering variance based criteria, a small region could be absorbed by a much larger one, whatever its gray level may be. Then the combined global variance of the main region (under consideration) and one of its eligible neighbours is calculated. The neighbour giving smallest value of variance after merging is selected as the "best neighbour" from the list of eligible neighbours of the main region. If a region does not have an eligible best neighbour then it cannot merge further, and it is removed from the list of valid regions. But it is possible that later this invalid region can qualify as a best neighbour of some other newly formed valid region.

F. Merging rows

When the region and its corresponding best neighbour are selected considering the above mentioned criteria in mind, it must be ensured that the variance of the selected region is less than the maximum variance value. The threshold value of the variance for the split and merge phase is the same. The regions are then merged and their list of neighbours is combined. The new neighbours list when merging region B to region A is the initial neighbours list of $A - \{B\} +$ the neighbours of B that are not neighbours of A. Hence the list of neighbours is updated for the new region(region R i.e. updated region A) so formed in the adjacency table. The size of the region R is the sum of the size of region A and B. A common label i.e. the label of the main region being merged (region A) is assigned to region R. Also the average grey level intensity of this region R is modified and updated.

The merging for a region is done in this manner till it can merge no further. The region becomes invalid when it cannot merge any more. Then another region is selected and is merged with its neighbours in the similiar way. This merging process continues till no more valid regions (that can merge further) exist.

III. Different approaches

A. Original method

Simple merging consists of selecting a region, merging it with its best neighbour, then selecting another region, and so on till no more valid regions exist. Good results were achieved using this approach but the time taken to perform the merging was much more than expected. Thus, we experimented several variants of this baseline method and compared them with respect to their computation time and quality of the segmentation (number of regions, final variance, etc).

B. One region at a time

In this method, one region is selected for merging and it is merged until it cannot merge any further. Then another region is selected and so on. This method gives almost the same segmentation quality but it with a much better execution time (that is improved up to 30 times).

C. Other considered modifications

- To increase the accuracy further,the modification made during 'best neighbour selection' was to check that if the 'main region' (under consideration) is also the best neighbour of its selected 'best neighbour'. So the merging was performed only if both the regions being merged were best neighbours of each other respectively. But this leads to highly selective merging and very few regions can be merged effectively. For instance, on the "cameraman" image with 1730 initial regions, the number of regions after merging were 1603 (instead of 531 regions otherwise).
- An expensive aspect in the merging, is the updating of the neighbour list. To ease neighbour search and selection, this list needs to sorted. Then we can use merge sort when performing the actual merging. An initially considered solution was to initially sort all the neighbor lists. The associated execution time was important, and we chose instead to perform the sorting "on demand", only for the case where the list needs to be merged. This leads to a dramatic reduction in the execution time (54 ms vs 1849 ms for the merge time for the "cameraman" image)
- A trick which has been used to decrease the computational merging time further is to modify the random selection of the main region in order to improve the execution time. In our method, the list of regions is written in the adjacency table and then random selection is performed by picking any region from this table. This table can be updated at any time during the merging process when a region is removed. But the problem is that it will require many useless copies and moves in the table. So instead we choose a "lazy" scheme, where the table is not modified but a new parallel list of all the regions is maintained in the beginning. Each time a region is picked randomly from this list, its validity is checked. If it is valid, it gets selected. But if it is invalid, then it is swapped with the last element of this list and the size of the list is decreased by one. In this way, this new list gets modified each time an invalid region is picked from the list so as to decrease the number of invalid regions in it and gradually increase the probability of picking out a valid region being picked out during random selection. This optimization makes the selection process faster. The same amount of merging is done in 36 ms instead of 54 ms for cameraman image.

D. Test image segmentation results

Three classical images have been used for benchmarking: cameraman, image and peppers. The computer used is a PowerPC G5 running at 2.5 GHz using gcc 4.0.

Figure 4 shows some segmentation results for several images. We display the original image, the segmented image (obtained by replacing pixel value in every region by the region average gray level) and the boundary image after merge for different thresholds of variance. The visually good results were obtained for a variance threshold of 30–40 for the given examples. The associated numbers represent the number of regions, before and after merging.

We can see that, despite some artifacts that are inherent to the method, mostly when dealing with regions with oblique frontiers, the overall segmentation is quite correct and can produce in real time a good starting point for a localization or recognition task.

IV. Graphs

A. Variance vs total Time

Figure 5 presents the total computation time for the benchmarked images for variance threshold range of (20-70). We can remark that the computation time lies in the range 20–45 ms leading to a possible implementation in real video framerate.

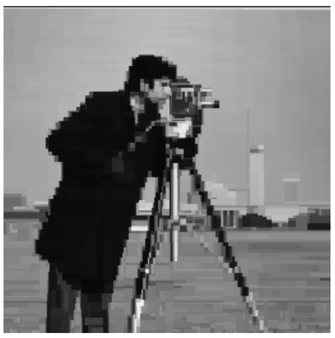
B. Detailed analysis of the execution time

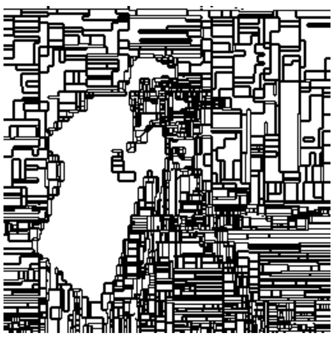
Figure 7 presents the detailed contributions of the different passes of the algorithm for the *image* image for different variances. We can notice that the split time is in general negligible (around 5 ms), and the most important contribution is the initial adjacency table construction, and the actual merging step. This later time increases with the variance, as the the deeper merging performed supersedes the smaller number of regions produces at the initial image slitting.

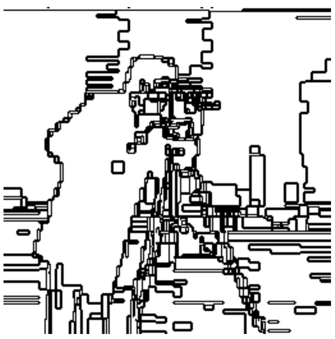
C. Variance vs Number of regions

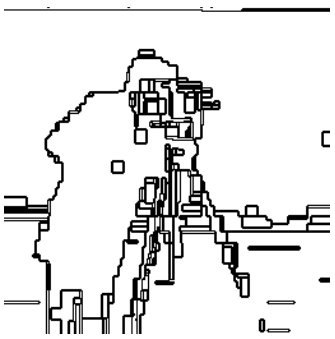
For all images (Fig. 6) the number of regions decrease with increase the value of variance threshold because greater the value of the threshold, more the number of regions will be merged.

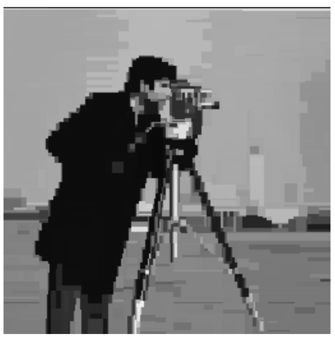
(a) Cameraman


(b) After optimal split(var -30)

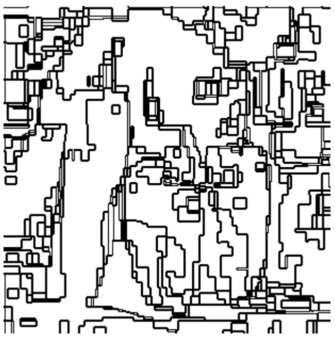

(c) Boundary image after merge, var-15,nb(2233,1528)

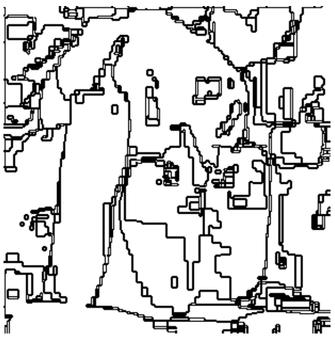

(d) Boundary image after merge, var-30,nb(1730,643)


(e) Boundary image after merge, var-50,nb(1441,230)

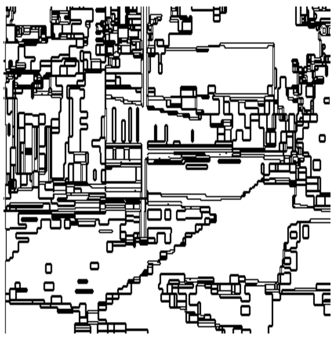

(f) Final segmented image(var 30)

(g) Peppers


(h) Boundary image after merge, var-40,nb(2039,571)


(i) Boundary image after merge, var-65,nb(1741,168)

(j) Image


(k) Boundary image after merge, var-35,nb(2631,775)

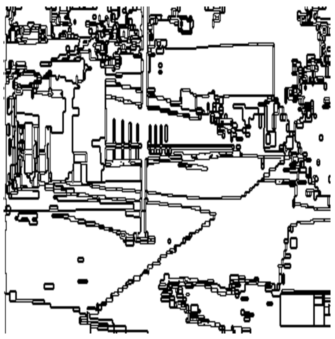

(l) Boundary image after merge, var-60,nb(2078,380)

Fig. 4. Examples of segmentation results

D. *Table*

For a variance threshold varying between 20 to 70, the average time and standard deviation is computed for each image in this table.

	cameraman	image	peppers
Average time (ms)	30	45	38
Standard deviation	8.0	6.8	6.3

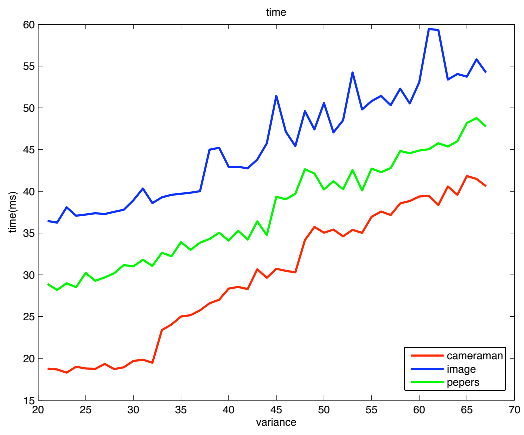


Fig. 5. computation time for the benchmarked images

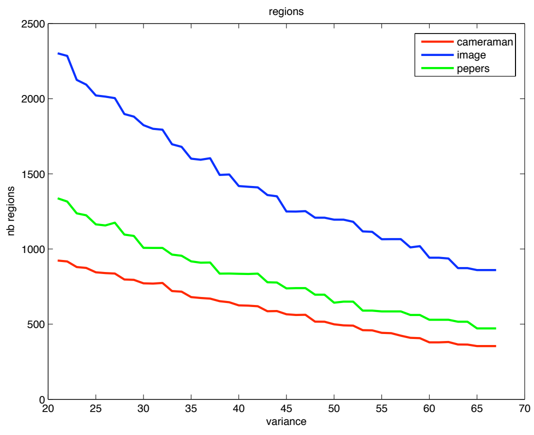


Fig. 6. number of regions for the benchmarked images

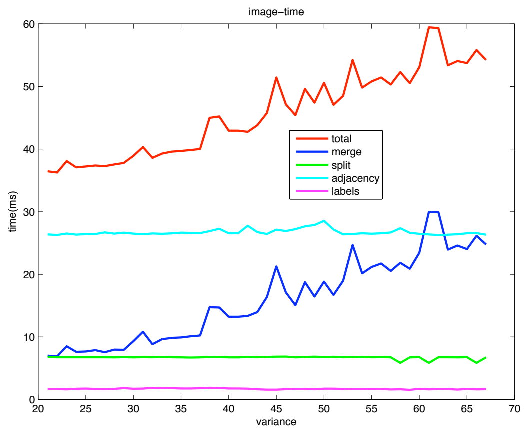


Fig. 7. computation time for *image* example

V. Comparison with Horowitz Pavlidis algorithms

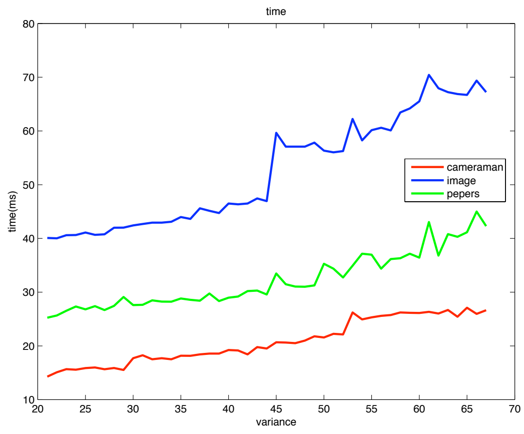


Fig. 9. computation time for *HP2* algorithm

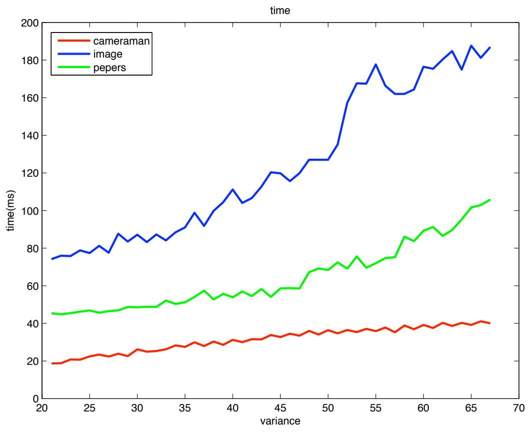


Fig. 10. computation time for *HP4* algorithm

As we can see (Fig. 9), HP2 is the only real-time algorithm (<40 ms). HP4 provides better split (Fig. 8) but is twice slower (Fig. 10) than HP2. finally the Optimal Split & Merge is as fast as HP2, but with better split & merge stages.

VI. Conclusion

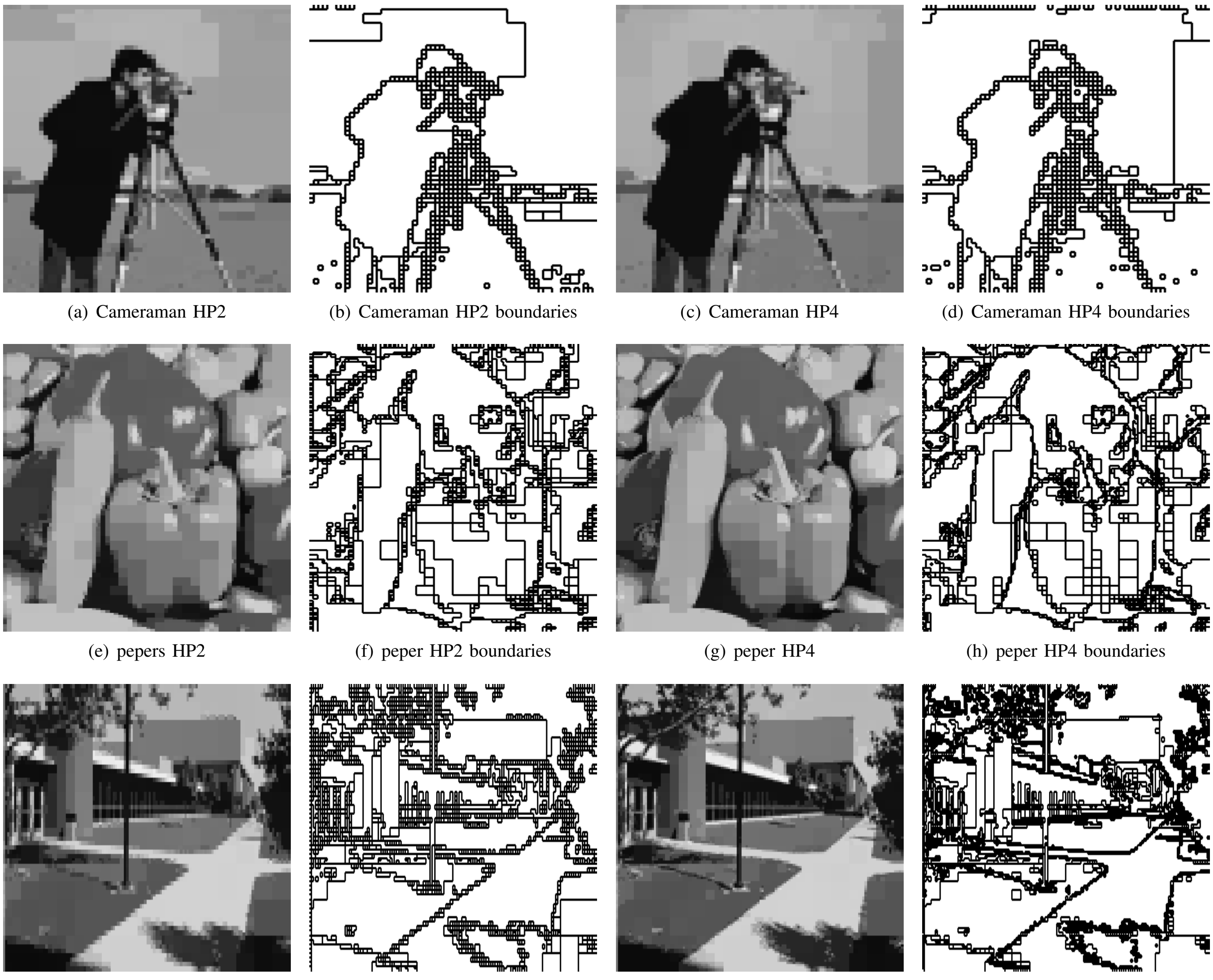
This paper has presented a real-time merging phase of a *real-time* segmentation scheme suitable for time critical applications. As it relies on an initial optimal split of the image, the quality of segmentation is much higher that most others methods based on classical split and merge, and it can be used for fast pre-segmentation purposes for localization or tracking. The quality of the segmentation can be further improved by postprocessing steps (for instance the segmentation done using successive steps[3]). Presently considered extensions are also to extend the method to color images.

References

[1] S. Horowitz and T. Pavlidis, *Picture segmentation by a tree traversal algorithm*, Journal of the ACM, 23:368388, 1976.

[2] Alain Merigot, *Revisiting image splitting,* 12th IEEE International Conference on Image Analysis and Processing (ICIAP 2003), 17-19 September 2003, Mantova, Italy, pp. 314–319.

[3] Andre Gagalowicz et Olivier Monga, *A new approach for image segmentation*. In Proceedings, Eighth International Conference on Pattern Recognition, IEEE Publ. 86CH2342-4, pp. 265–267, Paris, France,1986.


(a) Cameraman HP2 (b) Cameraman HP2 boundaries (c) Cameraman HP4 (d) Cameraman HP4 boundaries

(e) pepers HP2 (f) peper HP2 boundaries (g) peper HP4 (h) peper HP4 boundaries

(i) image HP2 (j) image HP2 boundaries (k) image HP4 (l) image HP4 boundaries

Fig. 8. Examples of Horowitz Pavlidis (HP2 & HP4) results