Region Analysis of Abdominal CT Scans using Image Partition Forests

Stuart Golodetz
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
stug@comlab.ox.ac.uk

Irina Voiculescu
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
irina@comlab.ox.ac.uk

Stephen Cameron
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
cameron@comlab.ox.ac.uk

ABSTRACT

The segmentation of medical scans (CT, MRI, etc.) and the subsequent identification of key features therein, such as organs and tumours, is an important precursor to many medical imaging applications. It is a difficult problem, not least because of the extent to which the shapes of organs can vary from one image to the next. One interesting approach is to start by partitioning the image into a region hierarchy, in which each node represents a contiguous region of the image. This is a well-known approach in the literature: the resulting hierarchy is variously referred to as a partition tree, an image tree, or a semantic segmentation tree. Such trees summarise the image information in a helpful way, and allow efficient searches for regions which satisfy certain criteria. However, once built, the hierarchy tends to be static, making the results very dependent on the initial tree construction process (which, in the case of medical images, is done independently of any anatomical knowledge we might wish to bring to bear). In this paper, we describe our approach to the automatic feature identification problem, in particular explaining why modifying the hierarchy at a later stage can be useful, and how it can be achieved. We illustrate the efficacy of our method with some preliminary results showing the automatic identification of ribs.

Categories and Subject Descriptors

I.4.6 [**Image Processing and Computer Vision**]: Segmentation; I.4.10 [**Image Processing and Computer Vision**]: Image Representation

General Terms

Algorithms

Keywords

partition forest, hierarchy, region adjacency graph, Bayesian classifier, medical imaging, abdominal CT

CSTST 2008 October 27-31, 2008, Cergy-Pontoise, France

1. INTRODUCTION

The ability to segment medical scans automatically and then identify important features (such as organs and tumours) in them is an important precursor to numerous medical imaging applications, including 3D visualization [13], volume estimation [9], and automatic landmark-based registration. Segmentation, the process of partitioning an image into semantically-meaningful regions, is difficult to automate in the case of medical images because the shapes of interesting features can vary significantly from one image to the next. Moreover, the boundaries between features are often unclear: for example, whilst a radiologist can easily tell the difference between normal parenchyma and cancerous tissue, automatically tracing the precise boundary is a much more difficult task.

One approach to the problem initially segments the image using the watershed transform, a common image processing technique from the field of mathematical morphology [3]. This works by treating the image as a landscape, with the grey value of a pixel giving its height. The landscape is divided into valleys (one per local minimum in the image), thereby partitioning the original image. The watershed tends to seriously over-segment images, however, so a standard approach is to then use region merging to reduce the region count to the desired level. The waterfall transform [3, 10] is a multi-pass, hierarchical algorithm designed for expressly this purpose. It naturally produces a partition hierarchy of the image, in which each node represents a region of the image: see Figure 1. This is variously known as a partition tree [12], picture tree [2], or semantic segmentation tree [1] approach (although these differ somewhat in the details, they are all based on the same basic principle). Each node of the tree can be annotated with useful properties for its corresponding region, and these properties can then be used when searching the tree to classify a region as being a particular feature of interest.

This is an interesting overall approach which should work well provided that the features we're interested in really do correspond to individual regions in the tree; unfortunately this is not always the case. The problem is in how the tree is constructed: it is common to merge together regions which satisfy some similarity criterion (e.g. similar grey levels). This is a reasonable first approximation, but provides no opportunity to bring crucial anatomical knowledge to bear on the problem. In this paper, we propose to solve this problem by allowing anatomically-based modifications to the hierarchy at a later stage of the algorithm.

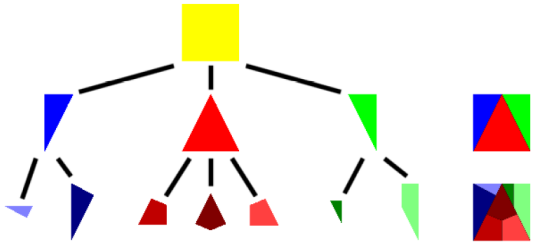


Figure 1: The partition tree for a simple image: the union of the regions at each level of the tree gives the whole image (see the squares on the right of the tree)

The layout of the paper is as follows: in §2, we define partition hierarchies and briefly explain how they are built. In §3, we describe an efficient annotation method that facilitates fast hierarchy updates by allowing region properties to be updated throughout the hierarchy with a minimal amount of recalculation. In §4, we present the actual hierarchy modification algorithms, explaining why they are necessary operations using examples from our work with abdominal CT scans. In §5, we briefly describe how to identify features from the partition hierarchy using a Bayesian classifier. In §6 we present our results and in §7 we outline our future work and conclude.

2. HIERARCHY GENERATION

We start our presentation of partition hierarchies by defining a few terms:

DEFINITION 1. *Given an image I, a* **partition** *$P(I)$ of I is a set $\{R_1, \ldots, R_r\}$ of non-empty and non-overlapping contiguous regions R_i. (So $\bigcup_i R_i = I$ and $\forall i, j \in [1, r], i \neq j \Rightarrow R_i \cap R_j = \emptyset$.)*

DEFINITION 2. *Suppose $P_f(I) = \{R_{f1}, \ldots, R_{fr(f)}\}$ and $P_c(I) = \{R_{c1}, \ldots, R_{cr(c)}\}$ are two partitions of I, then we write $P_f \sqsubseteq P_c$ iff for every region $R_{ci} \in P_c(I)$ there exists a region subset $S_i \subseteq P_f(I)$ such that $R_{ci} = \bigcup S_i$, and furthermore, $\forall i, j \in [1, r(c)], i \neq j \Rightarrow S_i \cap S_j = \emptyset$. (In other words, every region in P_c is the union of one or more regions in P_f, and no region from P_f is in more than one of these unions: thus, we say that P_c is a* **coarser partition** *of I than P_f.)*

DEFINITION 3. *Given a region set $R = \{R_1, \ldots, R_r\}$, a* **region adjacency graph (RAG)** *$RAG(R)$ of R is a graph with nodes R_i and edges represented by a function $w : R \times R \to \mathbb{R}^+$. There is an edge between any given pair of regions R_i and R_j iff $w(R_i, R_j) \neq \infty$, in which case $w(R_i, R_j)$ gives the weight on the edge.*

DEFINITION 4. *A* **partition hierarchy** *$H(I)$ of I is a sequence of n pairs $((P_1, RAG_1), \ldots, (P_n, RAG_n))$ such that each P_i is a partition of I, $P_1 \sqsubseteq \ldots \sqsubseteq P_n$ and $\forall i \in [1, n] \cdot RAG_i = RAG(P_i)$.*

Generating a partition hierarchy from an image is a bottom-up process. First, an initial partition of the image is somehow generated: in our approach, this is done using the watershed transform [3, 5, 11]. This initial partition forms the bottom layer of the hierarchy. Each higher layer is then generated by merging some of the regions in the layer below it. The choice of which regions to merge is made by whichever algorithm is in use: in our approach, we use the waterfall algorithm [5, 10]. The usual approach taken by such algorithms is to try and merge regions which are in some sense similar to each other (e.g. the difference between their mean grey values is tolerably small). The waterfall algorithm essentially works this way by performing a watershed-from-markers [4] process on the RAG of the current partition (or, more precisely, its minimum spanning tree): the effects of this depend on the weights assigned to the edges of the RAG, but these are usually some measure of the similarity between adjacent regions, e.g. the height of the lowest pass point on their shared boundary.

As well as generating the actual sequence of image partitions, the hierarchy generation process must construct a RAG for each partition/layer of the hierarchy. (These graphs provide the information needed to preserve region contiguity when modifying the hierarchy using the algorithms described in §4.) This can easily be done at the lowest layer of the hierarchy by iterating over the labelling produced by the watershed transform and adding edges to the RAG when adjacent points have differing labels. If the waterfall algorithm is being used, higher-level RAGs are then produced naturally from the initial RAG during the course of the algorithm; other algorithms may require alternative methods of RAG generation.

3. PROPERTY GENERATION

Having constructed a partition hierarchy, the next step is to annotate each node in the hierarchy with some properties of the region it represents. For the purposes of this paper, the properties used are area, mean grey value, grey value variance and elongatedness, but these are just illustrative, and many other properties can be used as well.

Our approach to property generation uses two phases. In the first phase, we use the original image, and its labelling produced by the watershed transform, to calculate the properties of the lowest-layer nodes. This is done by raster-scanning over both simultaneously and updating the region properties as we go. The second phase of the algorithm then generates the properties for higher-layer nodes by combining the properties of their children. As we will see in §4, this sort of approach facilitates fast updating of the tree, because the lowest partition never changes (thus the lowest-layer region properties never need to be recalculated) and updates to higher layers can be performed without going back to the original image.

The appendix details how the individual properties mentioned are generated.

4. HIERARCHY MODIFICATIONS

There are various operations we may want to perform on the annotated partition hierarchy. The most crucial of these is feature identification, whereby we mark a region in the hierarchy as a feature of interest. This process involves updating the hierarchy to reflect the fact that the marked region is no longer considered part of the image. We might also want to simply rearrange the partitions without identifying any features, e.g. by transferring a child from its parent to another node in the layer above, or by splitting a region into several pieces: the motivations for these are explained below. The first two operations have a lot in common: feature identification involves disconnecting a subtree and marking its root, whilst moving a subtree involves disconnecting it from its original parent and adding it to its new parent. The disconnection procedure is thus explained first.

4.1 Disconnecting a Subtree

The algorithm we have developed for disconnecting a subtree is best explained with an example. Consider Figure 3, which shows a simple striped image (a) and its corresponding partition hierarchy (b). Let us refer to regions using a 2D region locator of the form $(layer, index)$ (where *layer* is numbered from 0 at the bottom of the hierarchy), and suppose we want to disconnect region $(1, 1)$ from the hierarchy. The first step is to break the link between $(1, 1)$ and its parent, $(2, 0)$. We then recursively proceed down the subtree rooted at $(1, 1)$ and disconnect the region subset at that layer from the rest of the RAG for the layer: for instance, at layer 0 we disconnect the region subset $\{3, 4\}$ from the rest of the layer 0 RAG, but keep the edge actually connecting 3 to 4. The result is shown in (c). At this stage, the hierarchy (including the RAGs) is correct for all layers up to and including the layer in which the root of the subtree used to reside (1 in this instance). However, by disconnecting the subtree we have disconnected region $(1, 0)$ from region $(1, 2)$ in the image: merging them into region $(2, 0)$, as is currently the case, would now be an invalid operation. We therefore have to proceed recursively up the tree and rectify it. At each level, we have a set of child nodes and their purported parent. We determine the connected components of the children and ensure that each connected component has its own parent in the layer above: the properties of each parent are calculated from the children using property combination as described in §3. The RAG for the layer above is updated accordingly (this is done using the child layer RAG: parent A is adjacent to parent B iff one of A's children is adjacent to one of B's children). We then recurse, using the parent of the original purported parent as the new purported parent, and the union of the set of parents and the set of children of the new purported parent as the new set of children. This eventually produces a partition forest (e) as the result, something we expect since removing $(1, 1)$ from the image has divided the remainder into two pieces.

4.2 Feature Identification

Given the subtree disconnection algorithm, feature identification is now trivial: we disconnect the subtree, then mark the root of the subtree as being a particular feature.

(a) (b) (c)

Figure 2: The motivation for subtree moving: the red and yellow regions in (a) should be merged as they are both part of the vertebra, but the low grey level of the yellow region leads it to be incorrectly subsumed into another nearby region (b). Region moving allows us to move the sub-region back to its correct parent (c). The red region in (c) will now eventually be merged into a single region for the entire vertebra.

4.3 Moving a Subtree

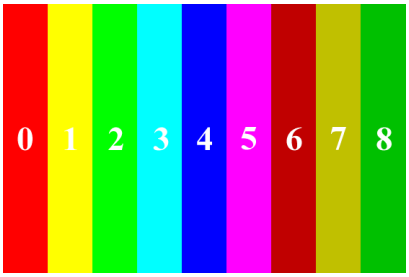
It can sometimes make sense to move a subtree by altering the parent pointer of its root to point to another node in the parent's layer. For instance, small pieces of a feature which are unclear on an image may get subsumed into adjacent regions with which they seem to share more in common (see Figure 2(b)). This can be corrected by transferring the sub-region to the parent which makes more semantic sense (see Figure 2(c)).

To do this, we first require that the piece being moved be adjacent to its new parent: this ensures that the resulting region will be contiguous. This can easily be checked using the child layer RAG (i.e. the one for the layer in which the piece resides) by testing whether the piece is adjacent to any of the children of the new parent. Provided this condition is satisfied, we move the subtree by disconnecting it from its existing parent (as described above) and adding it as a child of its new parent. In the process, the region properties for all the nodes on the path from the new parent up to the root of its tree (inclusive) will need to be updated using property combination.

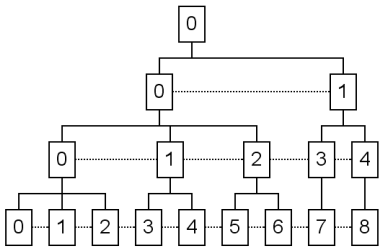
4.4 Splitting a Region

The final hierarchy operation we will describe is that of splitting an existing region. This involves taking a node at layer 1 of the hierarchy or above, and dividing it into a number of smaller (but still contiguous) regions, formed from mutually disjoint subsets of its children. This can make sense when too many regions were merged into one from one layer of the hierarchy to the next (the waterfall algorithm is a quickly-converging process, which is generally desirable, but occasionally problematic), or when it would be helpful to remove a single troublesome child region from its parent (see Figure 4).

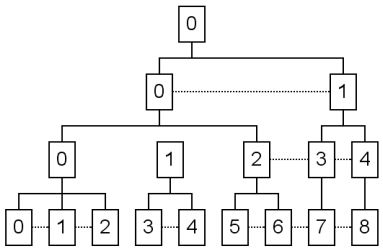
The process works by adding new nodes in the layer of the region to be split. To divide the region into n pieces, we add $n - 1$ new nodes, and change the parent pointers of the children to assign them to their new respective parents. We also update the region adjacency graph of the parent layer (i.e. the one in which the split region resides) accordingly. Finally, we update the region properties for the new parent nodes and the nodes on the path from them to the root of the split region's tree (inclusive). Figure 3(f) shows the result of splitting region $(2, 0)$ of the partition hierarchy in Figure 3(b) into two regions: one containing $(1, 0)$ and the



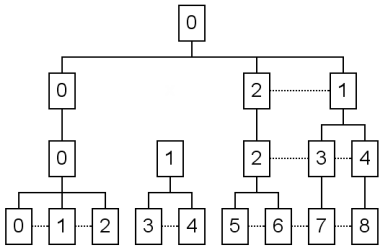
(a) A simple striped image



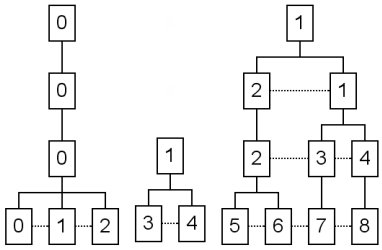
(b) A possible partition hierarchy corresponding to the image: solid lines are part of the tree, dashed lines are edges in the region adjacency graph for a particular layer of the tree



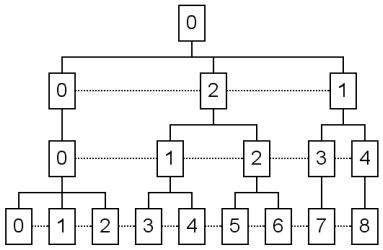
(c) Disconnecting the subtree rooted at $(1, 1)$: After the initial disconnection



(d) After rectifying the hierarchy up to layer 2 (the bottom layer is numbered 0)

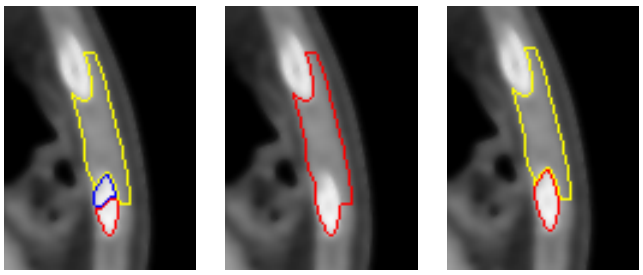


(e) The final result, after rectifying the hierarchy up to layer 3



(f) The result of splitting region $(2, 0)$ of the hierarchy in (b) into two

Figure 3: Hierarchy modifications



(a) (b) (c)

Figure 4: The motivation for region splitting: in (a), only the two small (red and blue) regions should be merged to form a rib, but the waterfall merges all three (b). Region splitting allows us to separate the overmerged region and extract the rib sub-region (c).

other $(1, 1)$ and $(1, 2)$.

5. REGION CLASSIFICATION

In §4, we discussed how the feature identification process updates the partition hierarchy; in this section, we explain how the actual decision is made to identify a region as a feature. Our initial approach has been to use a Bayesian classifier to quantify the extent to which a region resembles a feature of interest, based on some subset of the generated properties for that region. For instance, a rib classifier might be designed in terms of four properties $X_1, \ldots, X_4$: the area, max grey value, mean grey value and elongatedness of a region. (As a first approximation, ribs might be distinguished by having a relatively small area, a high max grey value, a relatively high mean grey value and a noticeable, but not extreme, elongatedness.) Given $\mathbf{P}(X_i|Rib)$ (i.e. a vector containing the probabilities of each value of X_i given that $Rib = T$ or $Rib = F$) for each X_i, and letting rib denote $Rib = T$, $\neg rib$ denote $Rib = F$ and x_i denote a specific value of X_i:

$$P(rib|x_1, \ldots, x_4) = \frac{P(rib)\prod_i P(x_i|rib)}{\sum_{r\in\{rib,\neg rib\}}\left[P(r)\prod_i P(x_i|r)\right]}$$

So far, the probabilities involved have been defined empirically, but we intend in future to derive them from a training set of images.

6. RESULTS

Our program automatically generates a partition hierarchy from each slice (as per §2), together with relevant properties for every region in the hierarchy. The user can visualize the hierarchy by navigating through a sequence of images representing its partitions. A feature corresponding to an individual region in the hierarchy can then be selected interactively. This process can be repeated for each desired feature. Figure 5 illustrates the result of selecting the aorta, kidney, liver, spleen and vertebra.

As already stated, however, the ultimate goal of our approach is to allow for automatic identification of features, with minimal user input necessary. Ribs were chosen as the

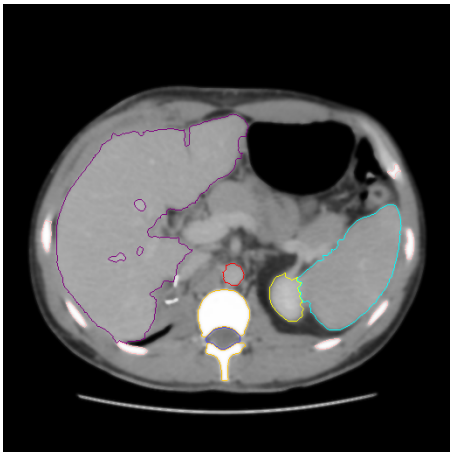


Figure 5: A sample result obtained by interactively selecting interesting regions in our partition hierarchy: aorta (red), kidney (yellow), liver (purple), spleen (cyan) and vertebra (orange)

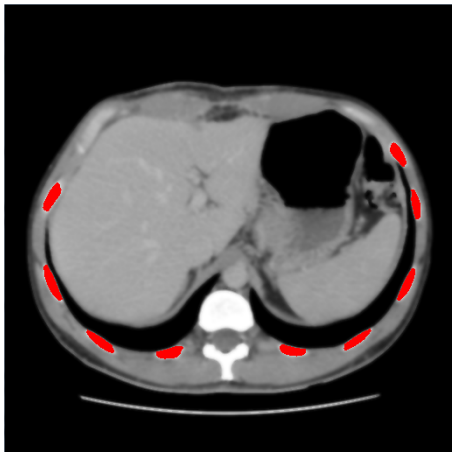


Figure 6: Ribs (shown in red) automatically identified from an abdominal CT slice using our algorithm

initial target of our algorithm because their high grey levels make them one of the easier features to identify on an image. For the actual identification we used a Bayesian classifier (as per §5) and have obtained promising results. Figure 6 shows the result of identifying the ribs in a CT slice. This is fully automatic and requires no input from the user.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we suggested an approach to automatic segmentation and feature identification in abdominal CT images, and presented some promising preliminary results. At this stage of the work, our focus has been on the new algorithms used.

Future work will aim to increase the level of automation, in particular by developing automatic identifiers for other features of interest, such as major organs and tumours. We plan to publish comprehensive statistical results for our method in due course. We have achieved a moderate level of success in automatically identifying ribs, and our experimentation with interactive region identification has suggested that many of the other interesting features in an image may be readily identifiable if the right region properties are used. To that end, we are currently investigating shape and texture description methods for regions [6, 7, 8], with a view to identifying sets of properties that can be used to determine which regions represent features.

We are also interested in looking at how to handle the case where both a region and one of its ancestors (e.g. its parent) in the hierarchy have been classified as being the same feature, and with similar classification probabilities. Picking the region with the higher probability may not always be the most appropriate choice.

Finally, we want to look into the possibility of sub-feature classifiers: as we have seen, two parts of a feature can be merged into different parents, and if we want to automatically modify the hierarchy to correct this, then we need a way of picking out sub-features which should have been merged together.

8. ACKNOWLEDGEMENTS

We would like to thank Dr. Zoe Traill, Dr. Andrew Protheroe, Mr. Mark Sullivan and Mr. David Cranston, of the Churchill Hospital, Oxford, for their invaluable help and support over the past year.

9. REFERENCES

[1] R. Al-Haj and Z. Al Aghbari. Semantic Segmentation Tree for image content representation. *International Journal of Virtual Technology and Multimedia*, 1(1):23–38, 2008.

[2] E. L. Andrade, E. Khan, J. C. Woods, and M. Ghanbari. Segmentation and Tracking using Region Adjacency Graphs, Picture Trees and Prior Information. In *IEE Visual Information Engineering (VIE'03)*, 2003.

[3] S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In *Proceedings of ISMM'94*, pages 69–76, 1994.

[4] P. Felkel, M. Bruckschwaiger, and R. Wegenkittl. Implementation and Complexity of the Watershed-from-Markers Algorithm Computed as a Minimal Cost Forest. *Computer Graphics Forum*, 20(3), 2001.

[5] S. Golodetz. Watersheds and Waterfalls (Parts 1 and 2). *Overload*, 83/84, February/April 2008.

[6] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson Education, 2nd edition, 2002.

[7] J. Iivarinen, M. Peura, J. Särelä, and A. Visa. Comparison of Combined Shape Descriptors for Irregular Objects. In *8th British Machine Vision Conference (BMVC'97)*, volume 2, pages 430–439, 1997.

[8] W.-Y. Kim and Y.-S. Kim. A region-based shape descriptor using Zernike moments. *Signal Processing: Image Communication*, 16:95–102, 2000.

[9] R. Lu and P. Marziliano. Liver Tumor Volume Estimation by Semi-Automatic Segmentation Method. In *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005.

[10] B. Marcotegui and S. Beucher. Fast Implementation of Waterfall Based on Graphs. In *Mathematical*

Morphology: 40 Years On. Springer Netherlands, 2005.

[11] A. Meijster and J. Roerdink. A Disjoint Set Algorithm for the Watershed Transform. In *Proceedings of the 9th European Signal Processing Conference (EUSIPCO '98)*, 1998.

[12] P. Salembier and L. Garrido. Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, April 2000.

[13] Z. Wu and J. M. S. Jr. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, July 2003.

APPENDIX

Property Generation Details

Notationally, we will represent a region in terms of its pixel set P, and $f(p)$ will denote the grey value of the original image at pixel p. Property combination will be illustrated by showing how to generate properties for a partitioned region $P = \bigcup_i P_i$ (where $P_i \cap P_j = \emptyset$) from those of its children P_is. For convenience, in what follows denote $F_{m,i} = \sum_{p \in P_i} (f(p))^m$ and $F_m = \sum_i F_{m,i} = \sum_{p \in P} (f(p))^m$.

The *area* for each lowest layer node can be trivially determined by adding 1 to the relevant count (referenced using the labelling) for each pixel of the image. Combining areas is likewise trivial: $|P| = \sum_i |P_i|$. Alternatively, $|P_i| = F_{0,i}$ and $|P| = F_0$.

The *mean grey value* for each lowest layer node can be determined by summing the grey values of pixels in the region (as specified by the labelling) during the raster scan, then dividing by the region area afterwards. Mean grey values can be combined using $m_P = 1/|P| \sum_i |P_i| m_{P_i}$; or $m_{P_i} = F_{1,i}/F_{0,i}$, $m_P = F_1/F_0$.

The *grey value variance* for a region with pixel set P is defined as $v_P = 1/|P| \sum_{p \in P} (f(p) - m_P)^2$. Alternatively, $v_{P_i} = F_{2,i}/F_{0,i} - (F_{1,i}/F_{0,i})^2$ and $v_P = F_2/F_0 - (F_1/F_0)^2$, and so the variances may be computed in a single scan.

Combining grey value variances may be achieved using the formula $v_P = 1/|P| \sum_i |P_i| (v_{P_i} + (m_{P_i} - m_P)^2)$, which can be derived as follows:

$$
\begin{aligned}
& \sum_i |P_i| (m_{P_i} - m_P)^2 \\
= & \sum_i F_{0,i} (F_{1,i}/F_{0,i} - F_1/F_0)^2 \\
= & \sum_i (F_{1,i}^2/F_{0,i} - 2F_1 F_{1,i}/F_0 + F_{0,i} F_1^2/F_0^2) \\
= & \sum_i (F_{1,i}^2/F_{0,i}) - 2(F_1/F_0) \sum_i F_{1,i} + (F_1^2/F_0^2) \sum_i F_{0,i} \\
= & \sum_i (F_{1,i}^2/F_{0,i}) - (F_1^2/F_0) \\
= & [F_2 - (F_1^2/F_0)] - \sum_i [F_{2,i} - (F_{1,i}^2/F_{0,i})] \\
= & |P| v_P - \sum_i |P_i| v_{P_i}
\end{aligned}
$$

Finally, the *elongatedness* of a region is defined as the ratio between the lengths of the major and minor axes of the minimum bounding ellipse of the region. These can be determined using a method known as principal component analysis. Firstly, for a region with point set P, we define the uv (central) moment of P as $\mu_{uv} = 1/|P| \sum_{(x,y) \in P} (x - \bar{x})^u (y - \bar{y})^v$. Note that in this, $\bar{x}$ and $\bar{y}$ are the x and y components of the region centroid, respectively. The covariance matrix of P is then defined as: $\begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$. The elongatedness can be calculated as the ratio of the square roots of the eigenvalues $\lambda_1 > \lambda_2$ of this matrix, $|\sqrt{\lambda_1}|/|\sqrt{\lambda_2}|$. Calculating the elongatedness throughout the partition hierarchy can be done indirectly by calculating the three moments μ_{20}, μ_{11} and μ_{02} at each lowest-layer node, combining the results to give the corresponding moments for higher-layer nodes, and then determining the elongatedness via eigen analysis at every node. Calculating and combining the moments uses a very similar approach to that employed for grey level variance, except that the update formula is now $\mu_{uv}^P = 1/|P| \sum_i |P_i| (\mu_{uv}^{P_i} + (\bar{x}_{P_i} - \bar{x}_P)^u (\bar{y}_{P_i} - \bar{y}_P)^v)$.