Content Based Image Retrieval with Scale Space Object Trees

David Dupplaw and Paul Lewis
Multimedia Research Group, Department of Electronics and Computer Science,
University of Southampton, Southampton, Hampshire, England, SO17 1BJ

ABSTRACT

Using recent advances in scale-space decomposition techniques, it is possible to transform images into trees in scale space, which represent the topology of regions within the image. We propose that complex objects can be found in the trees using a combination of subgraph isomorphism testing and conventional feature matching and hence provide a vehicle for achieving content-based retrieval and navigations for complex objects in images.

Keywords: Content Based Image Retrieval, Navigation, Scale-Space, Sieve, Graph Matching, Object Trees

1. INTRODUCTION

Ever since Vannevar Bush's early thoughts on the concept of Hypermedia[1] the idea has had appealing properties which many have sought to implement. However, only recently is the technology becoming able to achieve the necessary complex cross-referencing and indexing tasks, particularly on enormous data sets, like image and sound databases.

Many believe the final goal of media indexing functionality is automatic indexing and searching of media by their semantic meaning. However, the way this will probably be achieved is by using the low level content in the media (e.g. pixels and shapes in images) to create mid-level representations (e.g. objects in images) and eventually cross-reference with some knowledge-base of high-level "representation to meaning" functions, which could give meaning to low level content.

Indeed, the ability to extract simple, meaningful mid-level descriptions in terms of objects from images has been a long-standing goal of computer vision. It is only once objects are reliably extracted that research into the semantic indexing would be more accessible.

Our research into content based retrieval and navigation attempts to bring together some of the many techniques for content based retrieval in different types of media, into a single hypermedia system, called MAVIS (the Multimedia Architecture for Video, Image and Sound),[2] allowing cross-media navigation, and concept navigation and retrieval via a multimedia thesaurus. We are still, however, limited by the inadequacies of the low level feature matching of images, such as colour-layout,[3] simple outline shape or texture. We propose a mid-level extraction method for images based upon scale space matching and graph matching to achieve topological matching of images and object retrieval.

We propose a technique based on the Sieve.[4] The sieve uses morphology to decompose an image into scale space. A tree is built in scale space,[5] each node representing a region of the image which was removed at each scale of the sieve. Trees built from real-life images are generally far too complicated to match and a simplification of the tree can take place which will result in a simplification of the image. Because this manipulation is taking place in scale space, causality is not compromised. The simplified trees can be matched first by using subgraph isomorphism testing and then by feature matching at the nodes. Subtrees in the scale space tree may represent objects, which facilitates object retrieval by matching image selections represented as scale space tree subgraphs against a database of scale space tree representations of objects in the domain of interest. The topology matching would be rotation, scale, and transform independent. On the successful identification of similar structures in the image, other low-level matching techniques, such as colour, shape or texture matching, can take place on the identified nodes, to give a similarity match, allowing content based retrieval of images based on a specific object.

Send correspondence to : E-mail: dpd98r@ecs.soton.ac.uk
David Dupplaw: Email: dpd98r@ecs.soton.ac.uk
Paul Lewis: Email: phl@ecs.soton.ac.uk

2. THE SIEVE

The Sieve[4] is a morphological scale space operator developed by the University of East Anglia's School of Information Systems*. The Sieve has the effect of locating intensity extrema and "slicing off peaks and troughs" of a particular scale to produce regions.

One of the advantages of using the Sieve for scale space decomposition is that the contours of the image are not altered. Also, unlike Gaussian blurring, the Sieve preserves scale space causality - the axiom that states that no new extrema are created when moving to a coarser scale. Gaussian scale space decomposition techniques can break the scale space causality axiom by creating multiple extrema from a single extremum in anomalous situations. A well known example of this is that which Lifshitz and Pizer found[7] - the two regions connected by a thin isthmus. Gaussian blurring weakens the isthmus more than the regions which it connects, creating two maxima and a minimum in between. The Sieve is also fully invertible, meaning the original image can be re-produced exactly from the scale space decomposition.

The Sieve operates on connected set graphs.[8,4] The array of pixels in the image can be represented as a connected graph (4- or 8-connected for images) $G = (V, E)$, where V is the set of vertices and E is the set of pairs describing the edges. We define the $C_r(G)$ function to denote the set of connected subsets of G with r elements:

$$C_r(G, x) = \{\xi \in C_r(G) \mid x \in \xi\} \tag{1}$$

Morphological opening is achieved by taking the minimum value for every element of $C_r(G, x)$ and then taking the maximum of those minimums. The compact definition of opening, ψ_r, and closing, γ_r, is given in equations 2 and 3.

$$\psi_r f(x) = \underset{\xi \in C_r(G,x)}{max} \; \underset{u \in \xi}{min} \, f(u) \tag{2}$$

$$\gamma_r f(x) = \underset{\xi \in C_r(G,x)}{min} \; \underset{u \in \xi}{max} \, f(u) \tag{3}$$

Openings remove maxima in the signal, and closings remove minima. The $\mathcal{M}^r$ and $\mathcal{N}^r$ operators are defined respectively as an opening followed by a closing and, a closing followed by an opening both which remove maxima and minima. The results of the $\mathcal{M}^r$ and $\mathcal{N}^r$ operators are different and to ensure consistency in, for example, an image decomposition, only one of these should ever be used.

$$\mathcal{M}^r = \gamma_r \psi_r, \tag{4}$$

$$\mathcal{N}^r = \psi_r \gamma_r. \tag{5}$$

The regions that have been removed by the Sieve at any particular scale are called *granules*. A granule of scale s is removed by a Sieve of scale $s + 1$. These granules are moved into the *granularity domain* - a discrete scale space for the Sieve. Addition of all the granules in the granularity domain will return the original image - an example of the invertibility of the Sieve.

The output at scale r is defined as:

$$f_1 = \mathcal{Q}^1 f = f, f_{r+1} = \mathcal{Q}^{r+1} f_r \tag{6}$$

where $\mathcal{Q}$ is one of γ, ψ, $\mathcal{M}$, or $\mathcal{N}$. That is, a Sieve of scale r is a sequence of Sieves from scale 1 up to scale r. The complete sieve function, given by $\mathcal{CS}(f)$, is a function of Sieves of all scales that yield granules.

Because only the extreme part of the granules are removed, they are merged into the original image, the effect is to 'slice off' those peaks and troughs, always leaving larger regions in the image. This implies that all granules have 'parent' granules into which they are merged when they are Sieved. This is ideal for transferring into a tree structure where the links between the granules are more explicit, facilitating easier manipulation in the scale-space domain.

*Datasieve Ltd. hold the patent to the Sieve,[6] *http://www.datasieve.com/*

3. SCALE TREES AND OBJECT TREES

The fact that granules are continuously merged into 'parent' granules, means that they are ideal for transferring to a tree-structure[9] where each edge in the tree would represent a parent-child relationship between two granules. The nodes in the tree would represent the granules, and their properties. Granules from an image decomposition would have area (defined by their position in scale-space), shape, location and even colour and texture.

When a granule, g_s is removed from the image, it is merged into the 'parent granule' which it is adjacent to, g. This means that values of the pixels that it inhabited become the same as the larger granule around it. This is, of course, the same effect as removing the extrema. This makes the new granule a child of the parent granule: $child(g) = g_s$. This is made explicit with an edge between the two nodes in the scale-tree, representing the parent-child relationship.

The root of the scale-tree will be a single node in the centre of the image, which has branches to the next highest extrema in the image.

In certain cases, the scale-trees do not produce object trees. This often is the case for objects with holes in them. A good example in[9] shows that, for example, the hole in a teapot's handle can become associated with the teapot object, because the scale (area) of the hole is less than that of the teapot into which it gets merged. This type of error can be rectified by searching the subtrees for regions whose properties better match each other than the region it is associated with. In the case of the teapot, the colour saturation was used to decide that the hole really belonged to the background and not to the teapot. There is no general rule for dealing effectively with this type of problem, and the solution will depend upon the distinction of the object and the hole based on their properties. Bangham[9] calls the root node of a subtree which contains a true representation of an object a *grandmother node*.

It has been found[8,9] that the scale trees that are produced for real-world images have many thousands of nodes and would be very costly to store and manipulate. The problems with noise and aliasing in images causes lots of extra nodes to be produced in the scale-trees. Aliasing, and blurring, causes long branches in the tree, and noise causes many short sibling branches to be produced. Indeed, removing from the tree short branches which have very little change in properties to their parent, and using the invertibility of the granularity domain, acts as a very robust noise filter.[5]

Because graph matching is such a costly process, the size of the trees representing the images needs to be cut down dramatically before any matching can happen. The simplification of scale-trees is currently being researched[9] using statistical methods to remove the nodes from the tree. In particular, it is possible to remove nodes which are so like their parents that they are effectively redundant; this is the case for nodes created by small variations in noise caused by imperfections in the imaging system. This type of node removal has little noticeable effect on the rebuilt image. Long chains of nodes due to aliasing and blurring can also be collapsed into single nodes. This has the effect of sharpening the edges of the original image.

These types of reductions may cut the tree down to a hundred, or so, nodes. However, for matching, the trees may need to be cut down to a few tens of nodes, which will be merely a representation of the main features of the image.

Typically, there may be many objects in an image. A scale-tree representation of an image with multiple objects in it will have multiple grandmother nodes - root nodes of subtrees containing the representations of objects. A tree consisting of only grandmother nodes is consistent with the definition of an object tree, where object would fit the definition of a primary visual object in MPEG4.

We have implemented a decomposition technique which converts an image directly into the equivalent of a scale-tree, using a technique based on max-trees. It produces the same results much faster than the naive morphological approach, but not as fast as the patented implementation of the Sieve algorithm.[6]

An illustration of the sieve being used to produce trees can be seen in Figure 3 a) and b).

4. TOPOLOGICAL MATCHING

Trees are directed graphs, so it is possible to use graph matching techniques to find isomorphisms in trees. However, the matching of graphs is still an active research area and computational challenge. Because graph matching is NP-complete, special methods have been developed which use properties of the graphs, or properties of the required results to find results in polynomial time.

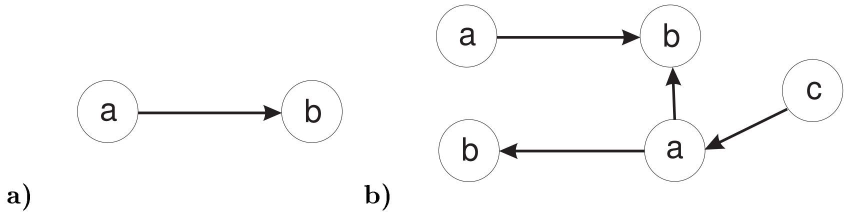

Figure 1. a) A model graph with two nodes which have different labels, b) A model graph with five nodes where some labels the same.

4.1. Graph Matching using Network Algorithm

Messmer and Bunke[10,11] give an algorithm which is based on the premise that the final application is to match a single input graph to a database of model graphs and to find the isomorphisms of the input graph in the model database. Of course, this is a classic retrieval procedure, and is perfect for what we are doing.

It uses a pre-processed network of disjoint, shared subgraphs of model graphs (e.g. scale-trees built from scale decomposed images) to find graph and subgraph isomorphisms in polynomial time. It is the compact network representation that achieves the fast detection. The method has been extended by Messmer and Bunke[10] to achieve inexact graph matching, so that distance measures can be found between graphs, based on the amount and cost of edits which need to be applied to convert one graph to a subgraph isomorphism of another.

Typically, the input graph would be a subtree of a scale-tree built from a decomposed query image, Objects in the scale-tree will have "grandmother nodes"[9] and the subtrees sourced by these grandmother-nodes would be ideal candidates for input nodes - effectively allowing retrieval of objects from the database. Selection of this grandmother-node could be semi-automatic or automatic if a sufficiently accurate node selection method can be found. Another alternative is to select a region in an image before hand, using semi-automatic segmentation tools ,[12] and then scale-decompose this into a tree, and use as an input graph to search in the database. It is acceptable to use a whole scale-tree as an input graph, although it would be less efficient than a simpler scale-tree.

Matches in the database will occur where the input image has similar structure and features to a subgraph stored by a network node in the database, which will be a subgraph of one or more model graphs.

The model database consists of a network of disjoint, shared subgraphs between all the models which have been inserted into the database. This gives a very compact representation of the models in the database. Figure 1 gives two model graphs which are used to create a network which is shown in Figure 2. It is possible to see the subgraph of node labelled 'a' joined to node labelled 'b' is shared by having an explicit subgraph in the network representing this.

Finding subgraph isomorphisms in the network is achieved by making each node in the network a matching procedure, and allowing the input graph to "filter" through the network. The method relies heavily on the labels of vertices within the graph to decide on whether there is an isomorphism or not. The labels in the scale-trees would consist of the properties of the region represented by that node in the scale-tree. This could be shape or colour for example and even the (x, y) coordinates. The label matching is an important part of the graph matching process, and it is foreseen that deciding on what constitutes label equality will play a major part in the success of the matching.

The network has four distinct types of nodes. The input-node, resides at the top of the network. There is only one input-node per network, and has no inputs, and has any number of outputs. The outputs are to l-vertex-checkers, which contain single vertices represented with particular labels. Outputs from two network nodes arrive at E-subgraph-checkers, and these contain subgraphs constructed from the disjoint contents of the two parents. The parents can be l-vertex-checkers (single vertices), or other E-subgraph-checkers (other subgraphs). The edges in the network between the nodes in the network, represent sets of edges which are used to join the two disjoint subgraphs which are stored in the nodes. At the base of the network are the m-model-nodes whose parent is a single E-subgraph-checker. They contain instances of the original model graphs which were input into the database. Every

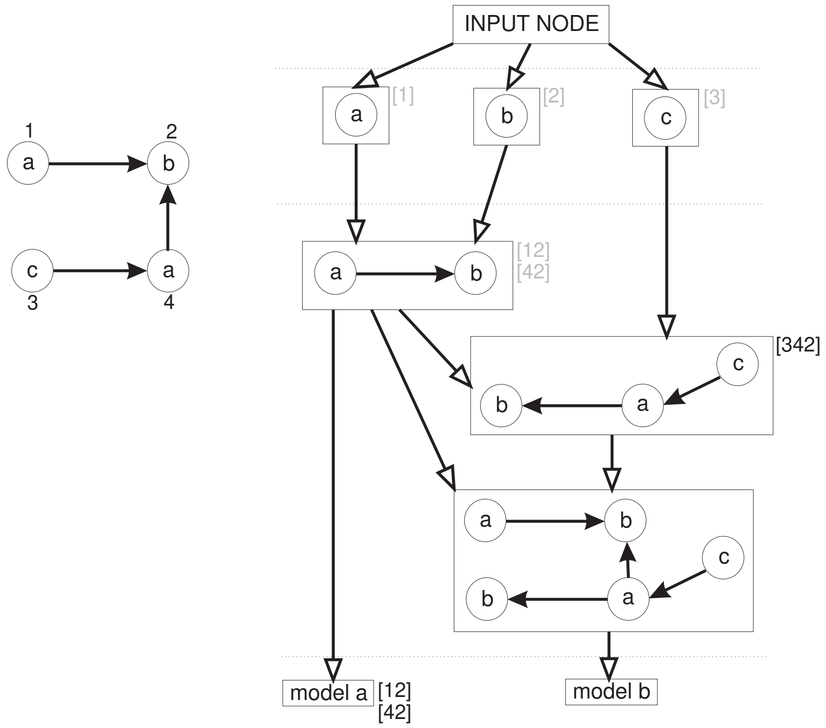

Figure 2. A network built with the graphs from Figure 1, a) and b). The dotted lines represent the different layers in the network: (from the top), the input node, the l-vertex-checkers, the E-subgraph-checkers, and the m-model-nodes. The graph on the left has been input into the network, and the local memories of the network nodes are shown. The grey memories are ones which are removed as the memories filter through the network, the black memories are those left at completion of the network algorithm. Three subgraph isomorphisms are found between the new input graph, and the two graphs in the database.

node in the network has a *local memory*, which is used to store the nodes from the input graph which match (using the label matching) the nodes in the subgraph which the network node stores.

Finding subgraph isomorphisms in the graph is achieved by initiating the network algorithm (NA) with the input graph. The l-vertex-checkers, in the network, each add to their local memory the instances of vertices in the input graph with labels matching the label they store, and pass the matching vertices on to their outputs. The E-subgraph-checkers check whether there are edges between any of the nodes in their parents, and if so, add an instance of the combined graph to their local memory, and remove the instances used from the parents' local memory. The m-model-nodes will only receive instances if the model graph is equal to, or a subgraph of, the input graph. The network node with the largest instance in its local memory, after termination of the algorithm, contains the largest subgraph isomorphism. Figure 2 shows the insertion of the graph into the network. Each node in the graph has been numbered for clarity. The instances of the input which match the subgraphs at the network nodes are stored in the local memory and passed to the outputs. The grey memories show the instances as they filter through the network. The black memories show the instances which are available at the termination of the network algorithm, and they represent the subgraph isomorphisms of the input graph, in the model graphs. So model graph, shown in Figure 1a), is a subgraph isomorphism of the input node of nodes 1 and 2, and nodes 4 and 2. Tracking through the network will find which model nodes these subgraphs belong to.

The label matching is an important part of the graph matching algorithm and is discussed in section 4.3.

One problem with the network is that it doesn't represent all possible subgraphs of the model nodes: for example, Figure 2 does not have an explicit subgraph representing the node labelled 'c' joined to the node labelled 'a'. This is due to the way in which the matching algorithm picks instances to merge. This is where the use of the inexact network algorithm will improve results, by changing the label of an instance between 'c' and 'a' and adding a cost, the subgraph testing can continue.

4.2. Graph Matching using Inexact Network Algorithm

The inexact graph matching is an extension to the exact graph matching network algorithm.[10] If the recall for the exact network is too low, even after variation of the thresholding levels on the feature matching, then the inexact matching should be invoked. With the inexact algorithm, a subgraph isomorphism between two graphs, G and G', consists of a sequence Δ of edit operations that are applied to G to create a subgraph isomorphism between $\Delta(G)$ and G'. An inexact subgraph isomorphism is optimal if there is no other sequence Δ' of edit operations that create a subgraph isomorphism between $\Delta'(G)$ and G', and has lower costs than $\Delta(G)$.

For this algorithm, there are defined five different edit operations which could occur to create a subgraph isomorphism from one graph to another:

- Substitution of a vertex or an edge
- Deletion of a vertex
- Deletion of an edge
- Insertion of a vertex
- Insertion of an edge

Each instance in the network then acquires a cost which gets cumulated as it propagates through the network. Those instances with the lowest cost at the end of the network algorithm are those which match best.

For the inexact algorithm to add costs, the nodes in the network have to be altered to affect the instances for cost. So, the l-vertex-checkers are altered so that reception of a vertex v with label k will cause the instance to gain the cost $c((v,k) \rightarrow (v,l))$ - a cost for changing the label k to the label l. $C(t)$ is defined as the cost for instance t. For vertex insertion and deletion to occur, a new 'dummy' vertex has to be defined. This takes the place of a deleted vertex, and is given the symbol \$. If the l-vertex-checker receives the dummy vertex \$ then the costs of the instance are set to the costs for the deletion of v, $c(v \rightarrow \$)$.

The functionality of the E-subgraph-checkers change in a similar manner. If they receive instances from their parents which are disjoint, or have the dummy vertex \$ in common, then they can be merged. Similarly, the costs are added for label substitution, edge addition and deletion, and the costs of the merged instance become $C(t) = C(t_l) + C(t_r)$, where t_l and t_r are the instances from the left and right parents.

However, the network algorithm itself requires modification to ensure that the algorithm does not become exponential in time. The way this is achieved is by having extra memories on the network nodes which only pass on one inexact instance if there are no exact instances. When an inexact instance is passed on it will be that with the lowest cost. This continues until an isomorphism of any kind is found.

The network algorithm with the altered network nodes is called ÑA, while the network algorithm that uses the extra memories is called INA.

4.3. Label Matching

The label matching is an important part of the graph matching algorithm. For the graph matching to work effectively the underlying low-level feature matching will have to be reasonably robust. Initially we are assuming a boolean expression for similarity, though later we plan to investigate using real valued similarities which will contribute to the success of the match. The boolean expressions can be achieved by using a threshold on the similarity measures of the features. The threshold will, of course, be dependent on the feature, but it will also be dependent on the application. Having the ability to alter the match thresholding on each of the feature matching techniques will allow the flexibility to alter the graph matching characteristics to suit any application. For example, if the colour of objects was more important than the shape, the similarity threshold for colour can be made more lenient than the shape match. To avoid extra calculation, the match statistics can be stored in the node for particular instances, so that once subgraphs are found, they already have the feature similarities available.

An illustration of the graph matching based on features can be found in Figure 3. The feature match at the nodes is based on shape, so that labels which have the same, or similar, shape can be considered a match. In this idealized case, the car becomes a subgraph in the image, and the tree another, thereby allowing any search based on the shape of a car or the shape of a tree, to find an isomorphism in the graph. The "whole image" node at the root of the decompositions plays an important part. Despite it having a shape representing the ratio of the image dimensions, and a will represent the salient colour of the image - which will often be the background colour. As can be seen from Figure 3 this node is often required to join subgraphs into a graph representing the image. It is foreseen that during matching this node may need to be treated as a special node, possibly containing a feature which simply states if it is a root node.

Another flexibility that is readily facilitated, is the ability to combine the characteristics with a logical AND or a logical OR. A certain application may require that the colour of the objects be a particular shade, and the shape is a particular form. In this case, the thresholding on the feature matching would be set so that only very similar features were classed as the same, and the results combined with a logical AND. A different application may require certain objects to be a certain colour, or a certain shape. In this case the best matches would still be those which were both the colour and the shape, because the match measures stored in the nodes would be highest. These combinations may be used at match time, to find similarly structured things, but with only certain features in common, or could be used during compilation of the network, to give a more compact, but less discriminating, network.

5. CONCLUSION AND FUTURE WORK

The use of scale space decompositions has been used by many to extract features from images [13–15] but we have not seen it used to aid content based retrieval. It is novel to use trees built from a scale space and match them using graph matching techniques. We are still investigating many aspects of this approach to matching images and objects, and have yet to perform controlled experiments.

The trees obtained from the scale space decomposition are still rather too large to be useful in a content based retrieval application. The tree reduction methods give more manageable trees by reducing chains of nodes, and removing nodes that have very small differences to the properties of their parent. The size of the trees created from real-world images will effect performance of the network algorithm, and we will have to experiment with the amount of reduction that is applied to the trees; reducing the trees too much will loose the descriptive ability for complex images, while not reducing them enough will create a performance hit on the graph matching network.

We plan on testing the system for object recognition as well as simple image similarity. Unlike many content based retrieval techniques, it is able to return a zero match measure - i.e. this image is totally unlike another - which would happen in the case that there were no l-vertex-checkers with similar features to input nodes. This means the recall of the system will have to be tested and many of the feature measures' thresholds investigated.

The inexact graph matching algorithm will allow a more usual similarity measure by applying costs to edits made in the graphs to force isomorphisms. This allows the more usual similarity measure to be returned from the matching function. The inexact network algorithm terminates when a subgraph isomorphism is found, with the lowest cost. However, it would be possible to continue to allow more inexact matches to filter through the network until some state is reached (such as the costs are too great). The recall of this system would be based on the number of graphs allowed to run through the network, and tests of how many to process will need to be made. This is also necessary to ensure that the inexact network algorithm does not become exponential in time complexity.

Based on the same ideas as the inexact network algorithm, it should be possible to filter the similarities of the feature matching through in a similar way to the costs of the inexact network algorithm. This avoids the need to use a threshold for each feature match at each network node, and would, instead, sort all instances at each node based on their feature similarity. The nodes with the highest similarity (lowest "cost") could be processed at each stage until a final isomorphism is found. Again, extra higher cost instances could also be filtered through to increase the recall of the system.

6. ACKNOWLEDGEMENTS

The first author is grateful to the Engineering and Physical Sciences Research Council (EPSRC) for the support of a research studentship. The second author is grateful to the EPSRC for the support of grant GR/L03446.

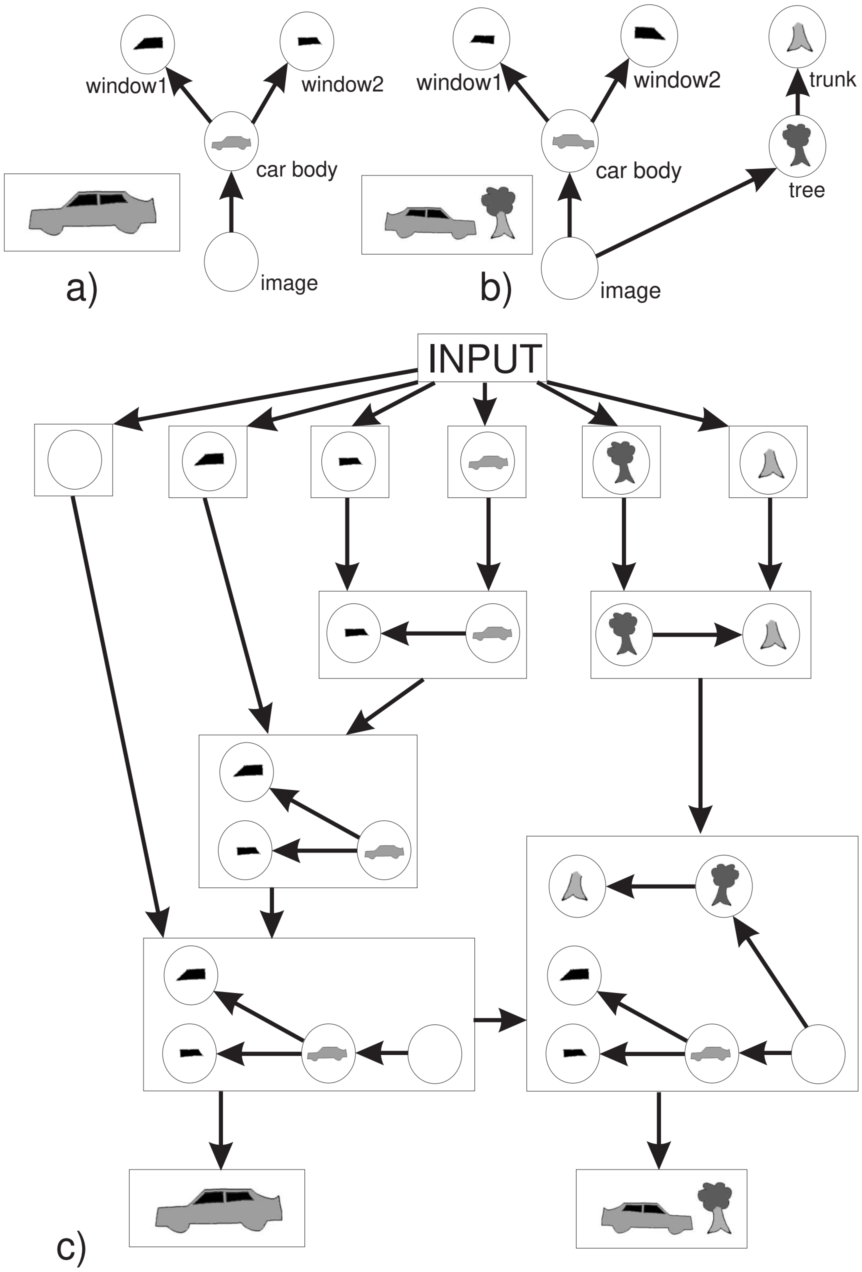

Figure 3. a) An illustration of a scale space tree built from the simple image; b) Another illustration of a scale space tree built from an image, where the image has subgraphs in common with the image in a); c) The network when both of these graphs from a) and b) have been inserted. It is possible to see the shared subgraph representing the car.

REFERENCES

1. V. Bush, "As we may think," *Atlantic Monthly* , June 1945.
2. M. Dobie, R. Tansley, D. Joyce, M. Weal, P. Lewis, and W. Hall, "A flexible architecture for content and concept based multimedia information exploration," in *Proceedings of the Challenge of Image Retrieval (CIR'99)*, February 1999. Also available from `http://www.mmrg.ecs.soton.ac.uk/publications/archive/dobie1999/`.
3. D. Dupplaw, P. Lewis, and M. Dobie, "Spatial colour matching for content based retrieval and navigation," in *Proceedings of the Challenge of Image Retrieval (CIR'99)*, February 1999. Also available from `http://www.mmrg.ecs.soton.ac.uk/publications/archive/dobie1999/`.
4. J. Bangham, R. Harvey, P. Ling, and R. Aldridge, "Morphological scale-space preserving transforms in many dimensions," *Journal of Electronic Imaging* **5**(3).
5. J. Bangham, J. Hidalgo, and R. Harvey, "Robust morphological scale-space trees," in *Proc. Noblesse Workshop on Non-Linear Model-Based Image Analysis (NMBIA'98)*, pp. 133–139, July 1998.
6. J. Bangham, "U.S. patent for Data processing method and apparatus," June 1999. Application Number: 875055.
7. S. Pizer and L. Lifshitz, "A multiresolution, hierarchical approach to image segmentation based on intensity extrema," *IEEE transaction on Pattern Matching and Machine Analysis* **12**(9), pp. 529–540, 1990.
8. J. R. Hidalgo, "The representation of images using scale trees," 1998. Master Thesis.
9. J. Bangham, J. Hidalgo, R. Harvey, and G. Cawley, "The segmentation of images via scale-space trees," in *Proc. British Machine Vision Conference 1998 (BMVC'98)*, pp. 33–43, September 1998.
10. B. T. Messmer and H. Bunke, "A network based approach to exact and inexact graph matching," Tech. Rep. IAM 93-021, Institut für Informatik, Universität Bern, Switzerland, September 1993.
11. B. T. Messmer, *Efficient Graph Matching Algorithms.* PhD thesis, University of Bern, Switzerland, 1995.
12. S. Perry and P. Lewis, "Novel image viewer providing fast object delineation for content based retrieval and navigation," in *SPIE Conference on Storage and Retrieval for Image and Video Databases VI.*, vol. 3312, pp. 436–445, SPIE, January 1998.
13. M. Fidrich and J.-P. Thirion, "Multiscale extraction and representation of features from medical images," Tech. Rep. 2365, Institut National de Recherche en informatique et en automatique, October 1994.
14. O. Olsen, "Multi-scale segmentation of grey-scale images," Tech. Rep. DIK U-96-30, August 1996.
15. J. Sporring and O. F. Olsen, "Segmentation by compression using linear scale-space and watersheds," in *Proceedings of Second International Conference on Scale Space Theories in Computer Vision*, September 1999.