Transforming Cluster-Based Segmentation for Use in OpenVL by Mainstream Developers

Daesik Jang[1], Gregor Miller[2] and Sidney Fels[2]

[1]Kunsan National University, Gunsan, South Korea
[2]Human Communication Technologies Laboratory,
University of British Columbia, Vancouver, Canada
[1]dsjang@kunsan.ac.kr, [2]{gregor,ssfels}@ece.ubc.ca

Abstract. The majority of vision research focusses on advancing technical methods for image analysis, with a coupled increase in complexity and sophistication. The problem of providing access to these sophisticated techniques is largely ignored, leading to a lack of application by mainstream applications. We present a feature-based clustering segmentation algorithm with novel modifications to fit a developer-centred abstraction. This abstraction acts as an interface which accepts a description of segmentation in terms of properties (colour, intensity, texture, etc.), constraints (size, quantity) and priorities (biasing a segmentation). This paper discusses the modifications needed to fit the algorithm into the abstraction, which conditions of the abstraction it supports, and results of the various conditions demonstrating the coverage of the segmentation problem space. The algorithm modification process is discussed generally to help other researchers mould their algorithms to similar abstractions.

1 Introduction

Research into computer vision techniques has far outpaced the research of interfaces (e.g. Application Programming Interfaces) to support the accessibility of these techniques, especially to those who are not experts in the field such as mainstream developers or system designers. Advances in the robustness of vision methods have led to a surge in real-world applications, from face detection on consumer cameras to articulated human body modelling for natural user interfaces. The algorithms capable of performing these feats are in the domain of experts, even if implementations are provided, due to the understanding required to: tune the parameters, which are often poorly documented and relate directly to variables in the mathematics of the method; form the input, which may include complicated templates for detection or pre-processed images (e.g. foreground-background separated); choose this method for the problem being solved - there are usually many methods, and it is a challenge even for experts to select the right algorithm given the conditions of the problem.

We argue that a simpler, higher-level interface can be provided to developers in order for them to utilise sophisticated vision methods. Our contribution in this

ACCV2012 #***

paper is an algorithm modified to fit a segmentation abstraction and a mapping of its specific algorithmic parameters to the abstraction's interface.

Developing an abstraction for computer vision is important for many reasons: 1) Developers may focus on their applications main task, rather than the algorithms; 2) Advances in the state-of-the-art can be incorporated into existing systems without re-implementation; 3) Hardware acceleration of algorithms may be used transparently; 4) The limitations of a particular platform can be taken into account automatically e.g. mobile devices may require a set of low-power consuming algorithms; 5) Computer vision expertise can be more readily adopted by researchers in other disciplines and general developers. If any abstraction is used to access vision methods, hardware and software developers of the underlying mechanisms are free to continually optimise and add new algorithms. This idea has been applied successfully in many other fields, notably OpenGL in graphics [1], but none has yet been successful within computer vision.

There has been a recent industry push to define standards for access to computer vision: the standards group Khronos have organised a working group to develop a hardware abstraction layer (tentatively titled CV HAL) to accelerate vision methods and provide simpler access mechanisms.[1] Khronos are proposing a layer beneath libraries such as OpenCV [2] in order to accelerate existing library calls (much like projects such as OpenVIDIA[2]).

We believe this abstraction layer has been targeted at too low a level to be useful for general developers. We propose an additional higher-level layer using a task-based abstraction to hide the details of algorithms, platforms and hardware acceleration from developers and allow them to focus on developing applications. The algorithm we present in this paper is tailored to an abstraction to provide developers with simpler access to segmentation results.

2 Related Work

Various surveys provide excellent overviews of the versatile approaches used for image segmentation. Shaw *et al.* surveyed important methods for segmentation based on intensity, colour and texture properties [3]. Skarbek *et al.* categorised various approaches more in depth focussing on colour segmentation [4] . Chan *et al.* showed some recent developments in variational image segmentation[5]. Zhangas surveyed unsupervised methods for image segmentatin[6]. Raut *et al.* added some modern approaches as well [7]. From these analyses we can summarise the important approaches of segmentation as follows:
Thresholding: These are generally used for greyscale images and are simple to implement [8]. Some methods use multi-dimensional histograms to extend this approach to include colour and texture properties for the segmentation [7].
Region: Region growing and region splitting-merging are the main procedures in this approach [9–11]. The region growing method groups pixels or sub-regions into large regions based on pre-defined criteria. Regions are grown from an initial

[1] http://www.khronos.org/vision
[2] http://openvidia.sourceforge.net

set of seed points, based on comparing neighbouring pixels' properties to that of the seed. Selection of seed points is therefore critical for colour images, and the result is highly dependent on these initial seeds.

Boundary: Edge detection is by far the most common approach for detecting meaningful discontinuities in grey level images [10]. In practice, edge-based techniques using sets of pixels seldom characterise an edge completely due to noise and non-uniform illumination which creates spurious intensity discontinuities. Hence edge detection algorithms need additional post processing by using linking procedures to assemble edge pixels into meaningful edges.

Graphing: The image is modelled as a weighted undirected graph [12]. Each pixel is a node in the graph, and an edge is formed between every pair of pixels. The weight of an edge is a measure of the similarity between the pixels. The image is partitioned into disjoint sets by removing the edges connecting the segments. The optimal partitioning of the graph is the one that minimises the weights of the edges that were removed. Shi's algorithm seeks to minimise the *normalised cut*, which is the ratio of the 'cut' to all of the edges in the set [13].

Morphology: One of the more stable techniques, the Watershed transformation considers the gradient magnitude of an image as a topographic surface [14]. Pixels having the highest gradient magnitude intensities (GMIs) correspond to watershed lines (which represent the region boundaries) - water placed on any pixel enclosed by a common watershed line flows downhill to a common local intensity minima. The method is initialised with markers to avoid over-segmentation due to noise and local gradient irregularities.

Clustering: Clustering for colour segmentation is a popular approach: it is especially effective when multiple features are engaged and one-dimensional methods like thresholding can not be applied. Colour within images is generally represented as multiple features, such as red, green and blue (RGB) or hue, saturation and intensity (HSI) [4]. Many techniques have been proposed in the literature of cluster analysis [10]. A classical technique for colour segmentation is k-means [15], extended to a probabilistic modelling using a fuzzy c-means algorithm [16]. There are various other approaches for segmentation via clustering, such as ISODATA (Iterative Self-Organizing Data Analysis Techniques) [10] and the mean shift algorithm [17, 18]. Connected-component labelling methods are used to compute the final segmentation based on the clusters [19, 20]. Clustering-based approaches are useful when the clusters of features are normal and easily distinguishable. If the features are cluttered among objects, this approach can not be guaranteed to give a good segmentation.

Automatic selection: Some automatic methods to select algorithms and parameters based on metrics or case-based learning have been tried recently. These approaches are meaningful in the sense that they can select an optimal algorithm and parameters adaptive to the characteristics of images to process. One methodology involves a generic framework for segmentation evaluation using a metric based on the distance between segmentation partitions [21]. Case-based reasoning was introduced to select an algorithm and parameters depending on the image characteristics [22]. The cases have image characteristics similar to

ACCV2012
#***

those of the current input image, and the segmentation parameters associated with the most similar case is applied to the input image. Yong *et al.* [23] proposed a simulation system designed to select the optimal segmentation algorithm from four candidates for synthetic images. Martin *et al.* [24] proposed a scheme to automatically select segmentation algorithm and tune theirs key parameters using a preliminary supervised learning stage. Nickisch *et al.* [25] proposed a new evaluation and learning method with user supervision.

While cluster-based methods for segmentation have drawbacks such as over-segmentation in the presence of high detail, they are extremely effective for isolating known regions. This is the case for developers designing applications with segmentation, where we envision the majority of use-cases are known in advance looking for a particular set of objects. We present a modified algorithm designed to accommodate a segmentation abstraction, which we present first.

3 Developer-Centred Segmentation

The central part of a segmentation framework provided to developers is a higher-level abstraction which hides algorithmic detail (the algorithm used and the parameters it uses) but still provides a powerful and flexible interface to segmentation results. We use a task-centred description for the interface, through which developers may describe the segmentation problem they need to solve.

For the abstraction we use a relatively simple definition of *segmentation*: producing a set of distinct regions (*segments*) within the image. We apply the concept of *properties* to measure distinctiveness. A property is measurable over a region of the image, which leads to an extensive list of possibilities, such as colour, intensity, texture, shape, contour, etc. Conceptually, a segment is bounded by a smooth, continuous contour, and is not dependent on pixels or any other discrete representation. Developers must specify at least one property to define the segmentation of the image: segment properties allow developers to decompose the image based on what they consider to be important to their problem, and provide us with the information required to produce a corresponding segmentation.

Each property is associated with a *distinctiveness* to allow the developer to define how distinct the segments should be with respect to that property. Due to the range of possible methods of segmentation, the term ‘distinct’ was chosen as the best abstraction of the conceptual meaning. This was in preference to terms such as *threshold* or *distance* which may be used in other methods but would not be applicable in all cases. The description also allows multiple properties for a single segmentation. Conceptually this will lead to segments which are distinct based on all specified properties. The advantage of the task-based description is the details of how this is performed are hidden from the developer, and so they do not need to take this into account when developing an application.

When defining the available set of properties we attempt to make sure each is orthogonal to the others, to avoid repetition in the description space and encourage completeness. Our eventual goal is to create a unified space for vision descriptions, to apply to all problems, which can be interpreted into algorithms

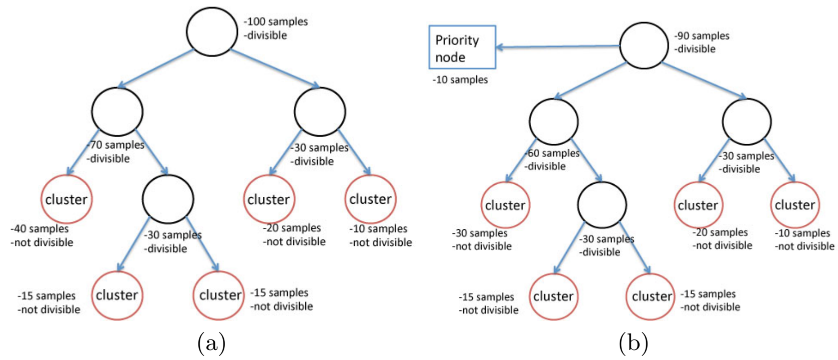


Fig. 1. Illustration of the use of a binary decision tree to create a set of clusters for segmentation. In (a), the initial feature space has 100 samples and is divided into two child nodes with 70 samples and 30 samples. The red nodes are determined to be cluster nodes since they are not divisible according to the end conditions. In (b), a priority node is added to capture use requirements for the grouping of similar pixels.

and parameters to provide the developer with a solution. The description space should be kept as small as possible while still maintaining a wide coverage to help minimise the complexity as the description language is extended.

The last aspect of the description is the use of *priorities*: the developer can define volumes in property space towards which the segmentation should be biased, which is useful in applications such as chroma-keying or skin-colour detection.

The properties and priorities together form what we define as the *requirements* of the segmentation. The last component we need to complete our description is *constraints*. Constraints introduce some additional complexity to the operation, because they are capable of overriding the distinctiveness requirement. The three constraints we provide are *size*, *quantity* and *regularity*. Size governs the final area of the segments, quantity the number, and regularity the level of variation allowed in the gradient of the segment's contour. Size and quantity are related and must trade off against one another; Regularity constrains the overall shape of the segments: a regularity of 0 does not constrain the shape at all and a value of 1 constrains the shape of every segment to be the same.

4 Transforming Cluster-Based Segmentation

Cluster-based segmentation is one of the most well-known and useful approaches for image segmentation. It is relatively simple to understand, practical for many use cases (especially when multivariate features such as RGB colours are used) and also benefits from good performance for general purpose segmentation. The major drawback is the difficulty for non-experts to understand how it works and the configuration required to achieve their required result. The parameters have a significant effect on the result of clustering and they should be carefully determined by experts to meet the requirements of each application. However, it is often difficult even for experts to match the parameters with the requirements

ACCV2012 #***

of diverse applications. In this section a cluster-based segmentation method for colour images is transformed to work with a developer-centred abstraction. The problem conditions of image segmentation can be described by developers instead of requiring expert knowledge of the algorithm and its parameters.

4.1 Method Overview

A cluster-based segmentation uses two main steps: feature-space clustering and region labelling. Various algorithms such as *k-means*, *fuzzy c-means* and ISODATA may be used to find clusters in the feature space. We use k-means with RGB colour, followed by connected component analysis for region labelling.

A conventional k-means algorithm is as follows:

1. Place K points in RGB feature space - these points represent the initial centroids of the clusters.
2. Assign each sample (RGB value of a pixel) to the cluster that has the closest centroid.
3. When all samples have been assigned, recalculate the centroids based on the newly assigned samples.
4. Repeat steps 2 and 3 until the centroids are static.

This produces a separation of the samples into clusters ready for post-processing: using the distance between two clusters as the metric, we can decide whether to sub-divide clusters or not (this is discussed further below).

One of the drawbacks of k-means is the requirement for a known number of centroids and the provision of each cluster with a good initial centroid. When the number of clusters is not appropriate for the input image, the segmentation can be over- or under-sampled. Variations of clustering such as ISODATA were developed to adjust the number of clusters by merging those that are similar, but it is still sensitive to the choice of initial centroids.

This weakness of k-means also makes it challenging to transform the method into a developer-centric framework. The clustering algorithm should adjust its parameters according to the description of segmentation to produce results satisfying the developer's requirements. The k-means algorithm is very rigid: its parameters do not neatly map to a developer-level description. To begin, we propose the parameters be adjusted as follows:

– Maximum number of clusters (K_{MAX}): determined according to the desired quantity of segments.
– Minimum distance of clusters (D_{MIN}): determined according to the desired distinctiveness of segments. If any pair of clusters are too close each other, they are not distinctive enough.

To adjust the algorithm to match our segmentation abstraction, a binary decision tree is combined with k-means to make these parameters adjustable. Instead of applying the k-means algorithm to the whole feature space, it is applied to a binary tree representing the feature space, starting from the root. This partitions the feature space of each node into two clusters. To illustrate: the root node contains the original feature space and it is divided into two clusters. Then the samples constituting the original feature space are divided into two subspaces based on the distance to cluster centroids. Two child nodes are generated with these two subspaces respectively and attached to the root node. K-means is applied to these two child nodes again in the same way. With this approach the initial K centroid points are no longer necessary: K can be determined by the framework through tree generation. Some conditions are required to stop the subdivision and control the size of the tree. The detailed algorithm for this new clustering method is as follows:

1. Make a root node with the samples of the original feature space.
2. For each node that is not classified as a cluster node:
   – Partition the node with k-means into two clusters.
   – Check the condition of the node with the provided parameters to determine whether it is divisible.
   –– If the node is divisible: divide the samples in the node into two subgroups and generate two child nodes.
   –– If the node is not divisible: it is classified as a cluster node.
3. Repeat step 2 until there is no node divisible.

Figure 1 shows the concept of using a binary decision tree for segmentation, and illustrates an example tree generated with this process. The conditions to determine the divisibility of a node use the following parameters:

– K_{MAX}: if the number of cluster nodes generated exceeds this parameter, all terminal nodes are marked as clusters and the process stops.
– D_{MIN}: if the distance between two clusters in a node is less than this parameter, then the node is determined not to be divisible and it becomes a cluster node.

Based on the identified clusters, a two-pass connected component labelling algorithm is used to generate segments (regions of the image) corresponding to the clusters. The two parameters of the clustering method are mapped to the description of segmentation in terms of properties and constraints. The details for this mapping are explain in the following section.

4.2 Parameter Mapping

The mapping of the parameters of the segmentation abstraction to our method are:

– *Distinctiveness*: The distinctiveness of produced segments is linked to D_{MIN}.
– *Quantity*: The quantity of segments to produce is linked to K_{MAX}.

When D_{MIN} is large, potentially divisible clusters may not be divided and the distinctiveness of clusters is decreased. For high distinctiveness, the parameter should be small enough to produce clusters with smaller gaps. K_{MAX} affects the quantity of segments: for large values the tree will contain more branches (and more clusters), therefore more regions are segmented. Table 1 shows the mapping between the clustering parameters and the segmentation description.

(a) (b)

Fig. 2. Sample images used for illustration of the results. Image (a) has dimensions 400×265, (b) has dimensions 553×720.

4.3 Priorities

The last property-related aspect of the description is *priorities*. These are supplied to let the developer specify volumes in property space which should bias the segmentation. This is important in the definition of boundaries: for example if a single-colour sphere is illuminated from one angle, the colour will have a gradient - the sphere's colour can be prioritised to segment the ball into a single region. With our segmentation method, we can accommodate priorities by inserting a new subspace defined by a volume in feature space; this can form a cluster and produce segments corresponding to the developer's requirements.

To implement *priorities*, a subspace corresponding to a developer-defined priority is expressed as a range of colours. This range is represented as a pair of RGB colour values and it constitutes a cubic subspace in the feature space. This subspace is represented as a special node in the binary decision tree and is attached to the root node. The samples which fall into the subspace are excluded from the root node so that the prioritized subspace is not considered for further clustering. Multiple priorities can be defined by adding additional priority nodes to the root node. Figure **??** shows the binary decision tree when a priority is defined, and an example of the clusters in feature space compared to the same space without a priority is shown in Figure 4.

5 Results

Our method was implemented (within the abstraction) in C++ and used OpenCV for utility functions; it was tested on a MacBook Pro Retina quad-core 2.6GHz with the images presented in Figure 2.

To illustrate the use of the abstraction-level *distinctiveness*, Figure 2 was segmented with *Low* (Figure 3(a), $D_{MIN} = 0.5$) and *High* (Figure 3(b), $D_{MIN} =$

Table 1. Parameter mapping from the developer-centred abstraction to the clustering algorithms parameters. This is an example set of numbers given an RGB feature space and approximate measures (*High, Medium, Low*) for distinctiveness and quantity.

Description	Clustering Parameter	Mapped Values
Distinctiveness	D_{MIN}	High : 0.01; Medium: 0.3; Low: 0.5;
Quantity	K_{MAX}	High : 20; Medium: 10; Low: 5;

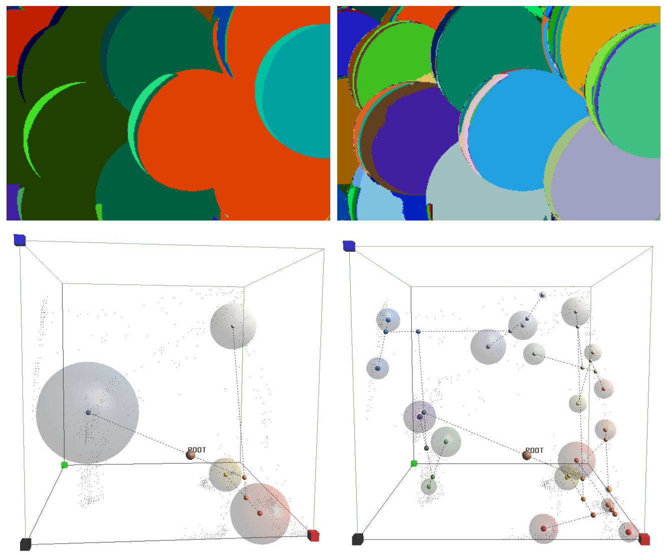
0.01) distinctiveness, both with $High$ set for *quantity* ($K_{MAX} = 20$). For *quantity* control, the results shown in Figure 5 have a Low quantity while leaving the distinctiveness constant (the images can be compared to the same distinctiveness with $High$ quantity in Figure 3(b) and Figure 4(a)). A priority-based segmentation result is shown in Figure 4(b), with the associated cluster tree with the priority volume (and cluster) shown in the top right of the feature space. A priority was given using a volume defined by the RGB range $(1.0, 0.0, 0.6) - (0.8, 0.2, 0.8)$ to hint to the segmentation method which parts of the feature space are important. The result shows the reddish regions of the image have been assigned the same label, a very different result from the over-segmented image in (a) with no priorities given. In all cases the method takes approximately one second to provide a result. Please note the implementation is not optimised to use accelerated hardware processing, and is intended as a proof-of-concept segmentation implementation to fit the abstraction defined in Section 3. The images demonstrate a close match between developer-provided parameters through the higher-level abstraction and the result produced by our segmentation method.

There is a relationship between D_{MIN} and quantity, and K_{MAX} and distinctiveness. The abstraction methodology is set up such that size, quantity and distinctiveness are related. To get very few segments, the developer could request a Low distinctiveness and a Low quantity - all parameters are used in the process, which provides the developer with greater control over the result.

6 Conclusions

We have presented a segmentation clustering algorithm which has been transformed to work with a developer-level abstraction, allowing non-experts access to sophisticated segmentation results. This has been achieved through the inclusion of a binary decision tree for creating clusters in feature space, mapping the abstraction description to the parameters of the method and modifying the clustering algorithm to allow segmentation biases to be included. Results demonstrate the clear mapping between the description a developer provides into the parameters used and the segmented images provided.

The abstraction and method will need to be modified to make it more clear to developers the impact on using distinctiveness and quantity as measures of segmentation (since they are linked); this may involve modifying the abstraction



(a) Low distinctiveness (b) High distinctiveness

Fig. 3. A feature-space visualization of a binary decision tree for clustering with the results shown above.

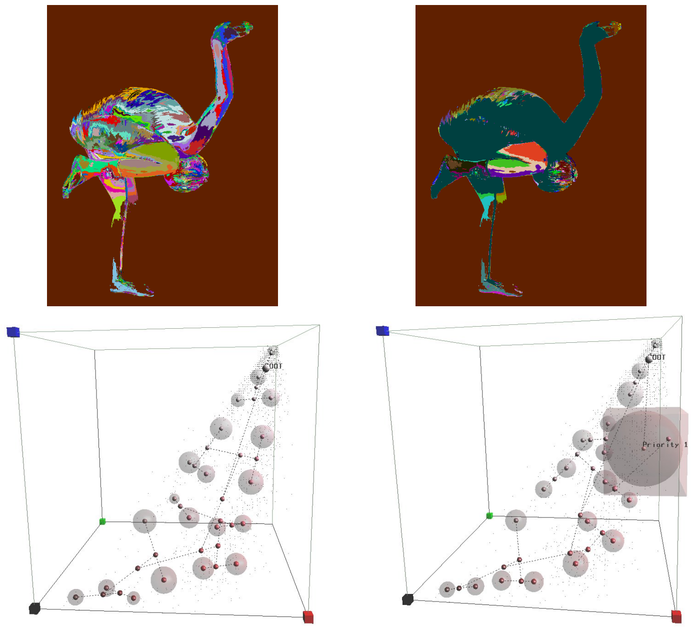
directly or making the results of using both for segmentation very clear, either with documentation or feedback from the abstraction framework.

We intend to transform other segmentation algorithms to fit the abstraction as provided, to include other properties such as texture, intensity and blur.

References

1. Shreiner, D., Woo, M., Neider, J., Davis, T.: OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition). Addison-Wesley Professional (2005)
2. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. 1st edn. O'Reilly Media, Inc. (2008)
3. Shaw, K.B., Lohrenz, M.C.: A survey of digital image segmentation algorithms. Final Technical Report ADA499374, Naval Oceanographic and Atmospheric Research Lab (1995)
4. Skarbek, W., Koschan, A.: Colour image segmentation - a survey. Technical report, Institute for Technical Informatics, Technical University of Berlin (1994)
5. Chan, T., Sandberg, B., Moelich, M.: Some recent developments in variational image segmentation. In: Proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, Springer (2005) 175–210
6. Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: A survey of unsupervised methods. Computer Vision and Image Understanding **110** (2008) 260–280

ACCV2012
#***



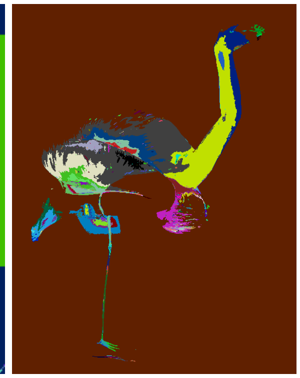
(a) High distinctiveness (b) High distinctiveness with priority

Fig. 4. A feature-space visualization of a binary decision tree for clustering, comparing the tree with (1) and without (2) priorities, and the results shown above.

7. Raut, S., Raghuvanshi, M., Dharaskar, R., Raut, A.: Image segmentation : A state-of-art survey for prediction. In: Proceedings of International Conference on Advanced Computer Control, New York, New York, U.S.A., IEEE Computer Society (2009) 420–424
8. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics **9** (1979) 62–66
9. Lucchese, L., Mitra, S.K.: Advances in color image segmentation. In: Proceedings of Global Telecommunications Conference, Berkeley, Calif, USA, IEEE Computer Society (1999) 2038–2044
10. Bow, S.T.: Pattern Recognition and Image Preprocessing. 2nd edn. CRC Press (2002)
11. Pavlidis, T., Liow, Y.T.: Integrating region growing and edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 225–233
12. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59** (2004) 167–181
13. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
14. Roerdink, J.B.T.M., Meijster, A.: The watershed transform: definitions, algorithms and parallelization strategies. Fundamenta Informaticae-Special issue on mathematical morphology **41** (2000) 187–228
15. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical

ACCV2012 #***

(a)



(b)

Fig. 5. The use of quantity (with *High* distinctiveness): *quantity* is set to *Low* ($K_{MAX} = 5$), and can be compared to the results in Figure 3(b) and Figure 4(a).

Statistics and Probability, Berkeley, Calif, USA, University of California Press (1967) 281–297

16. Kyoung-Bae Eumt, Joonwhoan Lee, A.N.V.: Color image segmentation using a possibilistic approach. In: IEEE International Conference on Systems, Man, and Cybernetics - SMC. Volume 2., New York, U.S.A., IEEE Computer Society (1996) 1150–1155
17. Comaniciu, D., Meer, P.: Robust analysis of feature spaces: Color image segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, U.S.A., IEEE Computer Society (1997) 750–755
18. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 603–619
19. Wang, W.: Color image segmentation and understanding through connected components. In: IEEE International Conference on Systems, Man, and Cybernetics. Volume 2., New York, U.S.A., IEEE Computer Society (1997) 1089–1093
20. Samet, H., Tamminen, M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. Transactions on Pattern Analysis and Machine Intelligence **10** (1988) 579–586
21. Cardoso, J., Corte-Real, L.: Toward a generic evaluation of image segmentation. IEEE Transactions on Image Processing **14** (2005) 1773–1782
22. Frucci, M., Perner, P., Sanniti di Baja, G.: Case-based-reasoning for image segmentation. Pattern Recognition and Artificial Intelligence **22** (2008) 829–842
23. Yong, X., Feng, D., Rongchun, Z., Petrou, M.: Learning-based algorithm selection for image segmentation. Pattern Recognition Letters **26** (2005) 1059–1068
24. Martin, V., Maillot, N., Thonnat, M.: A learning approach for adaptive image segmentation. In: Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS 2006), IEEE Computer Society (2006)
25. Nickisch, H., Kohli, P., Rother, C., Rhemann, C.: Learning an interactive segmentation system. In: Proceedings of the 7th Indian Conference on Computer Vision, Graphics and Image Processing, New York, USA, ACM (2010) 274–281