Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 485821, 9 pages
doi:10.1155/2008/485821

Research Article

Heterogeneous Stacking for Classification-Driven Watershed Segmentation

Ilya Levner, Hong Zhang, and Russell Greiner

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E8

Correspondence should be addressed to Ilya Levner, ilya@cs.ualberta.ca

Received 30 September 2007; Accepted 19 January 2008

Recommended by Sébastien Lefèvre

Marker-driven watershed segmentation attempts to extract seeds that indicate the presence of objects within an image. These markers are subsequently used to enforce regional minima within a topological surface used by the watershed algorithm. The classification-driven watershed segmentation (CDWS) algorithm improved the production of markers and topological surface by employing two machine-learned pixel classifiers. The probability maps produced by the two classifiers were utilized for creating markers, object boundaries, and the topological surface. This paper extends the CDWS algorithm by (i) enabling automated feature extraction via independent components analysis and (ii) improving the segmentation accuracy by introducing heterogeneous stacking. Heterogeneous stacking, an extension of stacked generalization for object delineation, improves pixel labeling and segmentation by training base classifiers on multiple target concepts extracted from the original ground truth, which are subsequently fused by the second set of classifiers. Experimental results demonstrate the effectiveness of the proposed system on real world images, and indicate significant improvement in segmentation quality over the base system.

1. INTRODUCTION

Pixel grouping and segmentation are two critical tasks in image processing and computer vision. If objects of the same predefined class are poorly delineated from the background or cannot be separated from one another, pixel grouping techniques can be employed for clustering the foreground pixels into objects. In order to separate two objects in close proximity to one another, the watershed algorithm [1] has been widely applied. Used within the unsupervised setting, the algorithm segments an image into a set of nonoverlapping regions. Embedded within the more general framework of mathematical morphology, the watershed algorithm considers a 2-dimensional gray scale image to be a set of points in a three-dimensional space, where the third dimension constitutes image intensity [2]. Segmentation is achieved by "flooding" the image topology, whereby water flows from areas of high intensity values along lines of steepest descent into regional minima (low intensity regions). In the end, individual watersheds or catchment basins of an image represent individual objects that are separated by the watershed lines.

Unfortunately, applying the watershed to the raw image rarely produces the desired result. The image is usually over-segmented into a large number of minuscule regions. As a result, several extensions have been proposed in order to produce more natural image segmentation (e.g., hierarchical watersheds or region split/merge [3]). Bar none, the most common remedy is to use markers [4, 5] for identifying relevant region minima. By setting marker locations as the *only* local minima within the watershed image, the number of regions can be automatically controlled. However, the process of finding a "good" set of markers can itself be problematic, nonintuitive, and ad-hoc.

To improve and automate watershed segmentation several machine learning approaches have been proposed. In [6, 7], a naive Bayes classifier was trained to identify and label pixel groups as internal markers. The discovered markers were then utilized, together with the color gradient magnitude of the image, by the watershed algorithm to identify and delineate colored cell nuclei. In [8], the classification-driven watershed segmentation (CDWS) algorithm furthered the notion of using machine learning to improve the watershed algorithm. Inspired by [6, 7], the CDWS utilized two distinct (sets of) classifiers trained to specialize in (a) marker identification and (b) object-background boundary delineation. In addition, rather than using the raw pixel values to train the classifiers, as was done in [6], the CDWS expanded the feature space by creating feature maps using standard image processing techniques, resulting in a very high pixel classification accuracy. Furthermore, the CDWS made additional use of the probability map produced by the object-background classifier. Rather than the conventional intensity or gradient magnitude image, the aforementioned probability map was employed as the topographic function within the watershed algorithm. Experimental results on gray scale and color image segmentation tasks demonstrated the effectiveness of CDWS on single and multichannel data.

CDWS proposed several novel ideas, including the use of ground truth manipulation, which is further explored in this paper. The original CDWS trained a pixel classifier h_{eroded} to detect markers. The "ground truth" for this objective was created by applying morphological erosion to the original pixel labeling ($\mathbf{L} \mapsto \mathbf{L}_{\text{eroded}}$). Figures 1 and 2 provide an example of this process. In this research, we further explore the use of ground truth manipulation by creating several new mappings (also shown in Figure 2). In addition to markers, the new target classes identify object boundaries that help in identifying markers, object regions as well as object boundaries. Subsequently, stacking, [9] is utilized to combine the output of the aforementioned classifiers in order to produce improved markers and object-background boundaries. The concept is called heterogeneous stacking and abbreviated as HS-CDWS.

Despite its success, the CDWS algorithm is not without its shortcomings. In particular, the original CDWS employed a set of manually engineered features, that, despite their generic nature, cannot work well in all potential domains. Furthermore, the need for explicit feature extraction demands a substantial knowledge of image processing and computer vision as well as domain expertise. To overcome this limitation, the second part of this research proposes using independent components analysis (ICA) for automating the feature extraction process. Unlike a fixed set of features, ICA enables the system to *learn* a feature set specific to the image domain at hand, and therefore allows for a greater degree of autonomy and flexibility.

The rest of the paper is structured as follows. Section 2 provides an in-depth overview of the CDWS algorithm from [8], and introduces the mathematical notation used throughout the article. Section 3 details heterogeneous stacking. Subsequently, Section 4 presents the feature extraction algorithm. Experimental results used to evaluate the efficacy of the proposed algorithms are provided in Section 5. The paper is concluded with final remarks and a discussion of future research directions in Section 6.

2. CLASSIFICATION-DRIVEN WATERSHED SEGMENTATION

2.1. Pixel classification

The particular data driven approach to image segmentation employed within CDWS attempts to learn a pixel classifier that assigns to each pixel the probability of belonging to a given class. Formally, let (i, j) index a discrete set of sites on a spatially regular $N \times M$ lattice:

$$S = \{(i, j) \mid 1 \leq i \leq N,\ 1 \leq j \leq M\} \tag{1}$$

for each input image $\mathbf{I}$ and the corresponding image labeling $\mathbf{L}$, let $\mathbf{I}(i, j)$ and $\mathbf{L}(i, j) \in \{0, 1\}$, respectively, denote the intensity values of image pixels and the corresponding (binary) labels. Throughout this paper, $\mathbf{L}(i, j) = 0$ labels the image pixel $\mathbf{I}(i, j)$ as background, while $\mathbf{L}(i, j) = 1$ denotes the pixel belongs to the target object class. The main objective is to produce a probability map $\mathbf{P}$:

$$\mathbf{P}(i, j) = p[\mathbf{L}(i, j) = 1 \mid \mathbf{I}(i, j)] \quad \forall (i, j) \in S \tag{2}$$

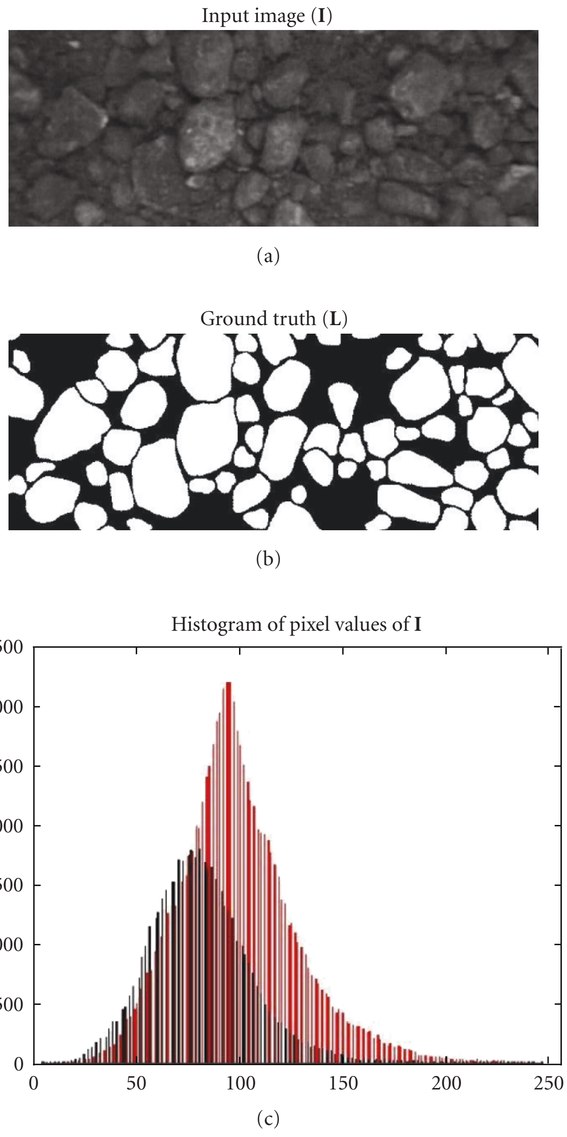

FIGURE 1: Image-based granulometry. *Top:* input image of a granulous material (in this case frozen oil sand ore) on a conveyor belt. *Middle:* ground truth image produced by a domain expert. *Bottom:* histogram of pixel intensities for each class.

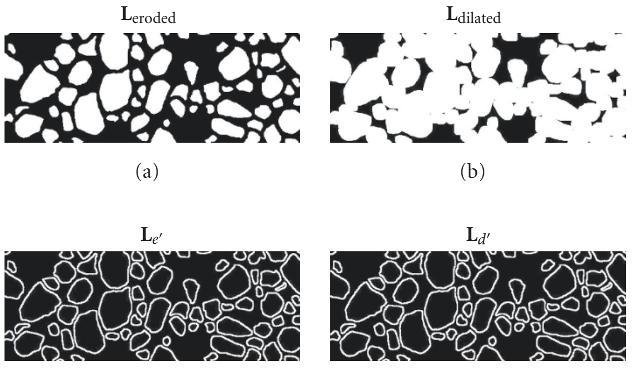

FIGURE 2: New target creation via morphological operations on the original ground truth (**L**).

with $p[\cdot]$ denoting the probability density function. To obtain the final image segmentation **L**, the probability map **P** is thresholded:

$$\mathbf{L}(i,j) = \mathbf{P}(i,j) > \tau \quad \forall (i,j) \in S. \tag{3}$$

The process in (2) treats individual pixels as i.i.d. (independent identically distributed). Unfortunately, this assumption is rarely satisfied in practice, since most nontrivial domains exhibit complex pixel interactions and dependencies. Therefore, simply using raw pixel values for classification in (2) results in very poor segmentation. (Otherwise, thresholding the input image at every pixel $\mathbf{I}(i,j) > \tau$ would produce the desired result. The histogram at the bottom of Figure 1 clearly demonstrates the practical shortcomings of this approach.) To overcome this problem, feature extraction techniques are needed to produce a set of feature maps describing local (and possibly global) image characteristics. The specific feature extraction method used in our research will be discussed in Section 4. For the moment, let $\mathbf{f}(i,j)$ denote the extracted feature vector at each lattice site (i,j). The probability map can now be conditioned on the feature vectors rather than just the raw gray scale values as follows:

$$\mathbf{P}(i,j) = p[\mathbf{L}(i,j) = 1 \mid \mathbf{f}(i,j)] \quad \forall (i,j) \in S. \tag{4}$$

The form $p[y = l \mid \mathbf{x}]$ in (4) defines an arbitrary binary classifier. As in [8], we model this class conditional using the generalized linear model (GLM) [10] and a logistic link function as follows:

$$p[y = 1 \mid \mathbf{x}] = \frac{1}{1 + e^{-\left(\omega_0 + \boldsymbol{\omega}_1^T \mathbf{x}\right)}} = h_{\boldsymbol{\omega}}(\mathbf{x}), \tag{5}$$

where $\boldsymbol{\omega} = \{\omega_0, \boldsymbol{\omega}_1\}$ are the model parameters, which can be estimated by maximizing the likelihood of the training data using standard nonlinear optimization routines, (The details of the optimization procedure can be found in [10, 11].) and $h_{\boldsymbol{\omega}}$ denotes the trained pixel classifier. From a Bayesian perspective, the model parameters $\boldsymbol{\omega}$ need to be integrated over using some prior distribution. However, this is usually intractable and is approximated in practice by learning a set of classifiers $\Omega = \{h_{\boldsymbol{\omega}_1}, \ldots, h_{\boldsymbol{\omega}_n}\}$, each optimized over a different subset of the training data. The outputs of each classifier are subsequently merged by uniform averaging as in bagging [12]:

$$H_{\Omega}(\mathbf{x}) = \frac{1}{n}\sum_{k}^{n} h_{\boldsymbol{\omega}_k}(\mathbf{x}). \tag{6}$$

Using (5) and (6) to model the probability map elements in (4), we get:

$$\begin{aligned}\mathbf{P}(i,j) &= p[\mathbf{L}(i,j) = 1 \mid \mathbf{f}(i,j)] \\ &= \frac{1}{n}\sum_{k}^{n} h_{\boldsymbol{\omega}_k}(\mathbf{f}(i,j)) \\ &= H_{\Omega}(\mathbf{f}(i,j)).\end{aligned} \tag{7}$$

To simplify the notation, we will refer to H_{Ω} simply as h in the remainder of the paper.

Provided relevant features $\mathbf{f}(i,j)$ have been identified, and the chosen machine learning technique, used to build the conditional probability model in (4), are capable of utilizing the extracted features, the outlined approach can achieve a high pixel classification accuracy. Unfortunately, even if the method exhibits good generalization performance, objects of the same class that are in close spatial proximity to one another will be merged together into a single connected component. Hence while the machine learned classifier may have a high pixel classification score, due to the unresolved object-object boundaries (i.e., under segmentation), the resulting object labeling can still be very poor.

2.2. Watershed segmentation

A popular approach to resolve object-object boundaries is to use region growing methods such as the watershed algorithm. However, to be effective the watershed algorithm requires object markers. Using ad-hoc rules to extract markers requires a priori knowledge of either (a) the number of objects within an image as in [4], (b) specific image properties, or (c) object locations (e.g., medical images registered to an anatomical template). In all cases, the parameters governing marker extraction tend to vary from image to image, again motivating the use of machine learning approaches for robust identification of object markers. In [6], the Bayesian marker extraction algorithm utilized a naive Bayes classifier in order to generate object markers. Unfortunately, since the classifier is trained on the ground truth delineating whole objects, the approach does not provide any constraints to ensure that only one marker per target object is extracted, nor that the extracted markers even lie within the object boundary. Naturally, one could threshold the probability map **P**, using a higher value for threshold τ in (3). As a consequence, precision will improve at the cost of recall, and thereby pixels that correspond (with higher probability) to object markers may be extracted. However, there is still no guarantee that the markers will be within object boundaries, nor that there will be a one-to-one correspondence between objects and markers. To improve the situation, in [8], a machine learning approach was proposed, that explicitly trained a marker identification classifier h_{marker}, on ground truth modified by morphological erosion. Let

$$\mathbf{L}_{\text{eroded}} = \mathbf{L} \ominus B \tag{8}$$

denote the erosion of the label image $\mathbf{L}$ by a suitably chosen structural element B. (For our experiments we used a disk with a radius of 7 pixels for the structural element.) The output of h_{marker}, denoted as $\mathbf{P}_{\text{marker}}$, is then given by

$$\mathbf{P}_{\text{marker}}(i,j) = p[\mathbf{L}_{\text{eroded}}(i,j) \mid \mathbf{f}(i,j)] = h_{\text{marker}}(\mathbf{f}(i,j)), \tag{9}$$

where h_{marker} is derived in the manner analogous to (7). To make the notational distinction more pronounced, we henceforth denote by h_{region} and $\mathbf{P}_{\text{region}}$ the classifier trained on the standard ground truth and the resulting probability map, respectively. The h_{marker} classifier is overly conservative (i.e., higher precision, lower recall) and produces superior object markers as compared to thresholding $\mathbf{P}_{\text{region}}$, using higher values of τ.

For topological surface needed by the watershed algorithm, again several options exist. The typical approach utilizes the gradient of the original image. However, since the probability maps themselves form a topological surface, the output of the machine learned probabilistic classifier can be utilized. Intuitively, the highest intensity values within $\mathbf{P}_{\text{region}}$ correspond to pixels with the highest probability of being part of the target class, hence using the inverted probability map $1 - \mathbf{P}_{\text{region}}$ can be advantageous because the aforementioned high-probability regions will be flooded first. To produce a topology amenable to the watershed algorithm, the inverted probability map $1 - \mathbf{P}_{\text{region}}$ is seeded with regional minima corresponding to marker locations extracted from the $\mathbf{P}_{\text{marker}}$ via hard thresholding (3).

3. HETEROGENEOUS STACKING

In [9], Wolpert introduced *stacked generalization*, which utilized the output of several base level ($\mathfrak{L}_0$) classifiers as inputs to the higher level ($\mathfrak{L}_1$) classifier, thereby improving classification accuracy. From a different perspective, one can view stacking as learning a gating function to control a mixture-of-experts [13], which in this case are the $\mathfrak{L}_0$ classifiers. The mixture-of-experts algorithms attempt to partition the input space into different regions or categories. In contrast, our approach explicitly partitions the output space and subsequently trains (a set of) classifiers on each newly created target concept. To combine these *heterogeneous* sources of information, we employ a second set of classifiers, analogous to stacking. To train the $\mathfrak{L}_0$ modules, we observe that even simple objects like the rocks presented in Figure 1 are not homogeneous, but instead contain several components that can be readily extracted by manipulating the ground truth in a manner analogous to producing $\mathbf{L}_{\text{eroded}}$ labels. Figure 2 presents four label images produced by applying the following morphological operations to the original label image $\mathbf{L}$:

$$\begin{aligned} \mathbf{L}_{\text{eroded}} &= \mathbf{L} \ominus B, & \mathbf{L}_{\text{dilated}} &= \mathbf{L} \oplus B, \\ \mathbf{L}_{e'} &= \mathbf{L} - \mathbf{L}_{\text{eroded}}, & \mathbf{L}_{d'} &= \mathbf{L}_{\text{dilated}} - \mathbf{L}. \end{aligned} \tag{10}$$

The transformations denote morphological erosion, dilation, and two difference operators resembling top-hat and bottom-hat operations. As in the original CDWS algorithm, $\mathbf{L}_{\text{eroded}}$ identifies object markers, while $\mathbf{L}_{e'}$ and $\mathbf{L}_{d'}$ identify inner and outer object boundaries, respectively. In turn, boundary information indicates where markers and object regions (i.e., $\mathbf{L}$) *cannot* be found. Hence these newly extracted target concepts are complementary to each other and to the original ground truth. Consequently, the $\mathfrak{L}_1$ gating network needs to fuse the output of $\mathfrak{L}_0$ classifiers together rather than select the output of a single base classifier as in defacto mixtures-of-experts algorithm. From this point of view, our work resembles ensemble learning algorithms, for example, bagging [12] and boosting [14], which are inherently cooperative in nature. However, these methods introduce diversity into the ensemble by resampling the training set as does stacked generalization. In contrast, we modify the label image $\mathbf{L}$ and otherwise keep the training set unchanged. *Random* label flips have been previously explored in [15–17]. Of course once the i.i.d. assumption has been made, as was done in the aforementioned references, there is nothing more "intelligent" one can do with the training data other than to try and regularize the learning algorithm via the aforementioned random label permutations. In contrast, image pixels, for any nontrivial domain, are definitively not i.i.d. (cf., Figure 1) and are, therefore, amenable to much more interesting label modification schemes. To the best of our knowledge, our research is the first to propose explicit and knowledge directed modification of the ground truth image.

Having defined all target concepts $\mathbf{L}_{\text{type}}$, where type $\in$ {region, eroded, dilated, e', d'}, the corresponding probability maps are created by generalizing (9) as follows:

$$\mathbf{P}^{\{0\}}_{\text{type}}(i,j) = p[\mathbf{L}_{\text{type}}(i,j) \mid \mathbf{f}^{\{0\}}(i,j)] = h^{\{0\}}_{\text{type}}(\mathbf{f}^{\{0\}}(i,j)). \tag{11}$$

Noting that this set of probability maps forms a multidimensional image, we simplify the notation by letting $\mathbf{P}^{\{0\}} = \{\mathbf{P}^{\{0\}}_{\text{type}}\}$. Recently, Ting and Witten [18] have empirically demonstrated that using the raw probability maps rather than the thresholded classification labels as input to $\mathfrak{L}_1$ classifier(s) improves performance. As our experimental results will demonstrate, for non i.i.d. data, one can go further and interleave feature extraction with learning to further improve performance. Once again, this effectively allows us to take advantage of the rich domain structure present within images and the resulting probability maps. Consequently, the second round of feature extraction can be implemented via the following mapping:

$$\mathbf{P}^{\{0\}} \longmapsto \mathbf{f}^{\{1\}}, \tag{12}$$

where $\mathbf{f}^{\{i\}}$ denotes the ith level of feature extraction. Subsequently, the extracted features can be utilized to train a set of $\mathfrak{L}_1$ classifiers $h^{\{1\}}_{\text{type}}$, where type $\in$ {region, eroded}.

The final labeling $\mathbf{L}^{\{\text{final}\}}$ can then be produced by creating a topology usable by the watershed algorithm from the probability maps $\mathbf{P}^{\{1\}}$ and applying the watershed algorithm. The process was described in Section 2. Within the stacking framework, the topology creation process can be viewed

$$\mathbf{I} \mapsto \mathbf{f}^{\{0\}} \xrightarrow{h^{\{0\}}} \mathbf{P}^{\{0\}} \mapsto \mathbf{f}^{\{1\}} \xrightarrow{h^{\{1\}}} \mathbf{P}^{\{1\}} \mapsto \cdots \mapsto \mathbf{P}^{\{\lambda\}} \mapsto \mathbf{f}^{\{\text{ws}\}} \xrightarrow{\text{ws}} \mathbf{L}^{\{\text{final}\}}$$

FIGURE 3: Generic set of mappings describing the process of HS-CDWS with $\lambda + 1$ levels. The last level represents the application of the watershed algorithm, abbreviated as ws.

as a feature extraction step mapping $P^{\{1\}} \mapsto \mathbf{f}^{\{\text{ws}\}}$, while the watershed process can be viewed as an unsupervised classifier. The heterogeneous stacking process (named, HS-CDWS) can now be succinctly summarized by a sequence of mappings presented in Figure 3.

4. $\mathfrak{L}_0$ FEATURE EXTRACTION

Currently, many different feature extraction approaches have been proposed in the literature, with texture features being most relevant [19–21]. Common descriptions of texture include: (a) cooccurrence matrices [22], (b) local binary patterns [23], and (c) random field methods [24]. In [8], the feature extraction resembled Viola's approach [25, 26], which utilizes a sequence of linear filters to produce the feature maps. In contrast, [8] used more general algorithms for extracting feature maps in order to compose a multichannel image $\mathbf{f}$, whereby each pixel vector $\mathbf{f}(i, j)$ corresponded to a single training/test sample. The large set of simple and redundant feature maps $\mathbf{f}_\alpha$, $\alpha \in \{1, \ldots, k\}$, was created with the expectation that the (logistic regression) classifier will weight each map according to relevance for a given task. Unfortunately, it is impossible to produce a single static set of features applicable to a large number of domains. To encompass an ever increasing set of domains, one must continuously add features. Inadvertently, this process increases computational complexity (both during learning and at runtime) and introduces unwanted feature interactions which in turn prevent logistic regression (and any classifiers expecting an independent set of features) from learning a correct set of weights $\boldsymbol{\omega}$. To overcome these problems, feature selection methods can be utilized in order to create a small set of independent features relevant to a specific task.

In contrast to the aforementioned manual feature design coupled with feature selection, we turned our attention to fully automated methods. The proposed approach removes the need for manual feature extraction altogether, by using independent components analysis (ICA) to automatically extract features from raw image patches [27]. In general [28], the ICA model represents data vectors ($\mathbf{x}$) as linear mixtures of latent feature vectors ($\mathbf{s}$):

$$\mathbf{x} = \mathbf{A}\mathbf{s} = \sum_k \mathbf{a}_k s_k, \tag{13}$$

where $\mathbf{A}$ is an unknown mixing matrix. For feature extraction, we are interested in finding the latent variables by applying the pseudoinverse of $\mathbf{A}$, denoted as $\mathbf{A}^\dagger$ to $\mathbf{x}$

$$\mathbf{s} = \mathbf{A}^\dagger \mathbf{x}. \tag{14}$$

Numerous ways of estimating $\mathbf{A}$ (or its pseudoinverse) have been proposed in the literature [29]. Most of the algorithms optimize some measure of statistical independence between the latent features $\mathbf{s}$, via gradient descent techniques.

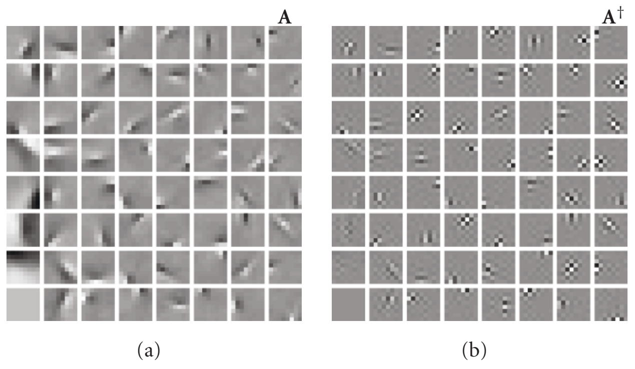

FIGURE 4: A typical result produced by ICA. *Left:* matrix $\mathbf{A}$ with each row reshaped into a patch. *Right*: matrix $\mathbf{A}^\dagger$ with each column reshaped into a patch representing a filter bank. The "optimal stimulus" for each filter is given by the visualization of the corresponding row in $\mathbf{A}$.

For images, each vector $\mathbf{x}$ represents a vectorized $n \times n$ image patch. Conveniently, the rows (resp., columns) of $\mathbf{A}$ (resp., $\mathbf{A}^\dagger$) can reshaped into image patches and visualized as in Figure 4.

Once the matrix $\mathbf{A}^\dagger$ has been learned, features can be efficiently extracted by reshaping the columns into filters, and subsequently convolving an input image with the newly created filter bank. (Typically, the input image is normalized by subtracting the mean and dividing by the standard deviation). Furthermore, the local mean is then subtracted from each $n \times n$ patch. The local mean normalization can be efficiently implemented via convolution as well.) We denote by $\grave{\mathbf{a}}_\alpha$ the filters created from $\mathbf{A}^\dagger$. The set of filters is denoted by $\Phi = \{\grave{a}_1, \ldots, \grave{a}_k\}$. Hence the feature maps $\mathbf{f}_\alpha$ can be produced via convolution by

$$\mathbf{f}_\alpha^{\{0\}} = \mathbf{I} * \grave{\mathbf{a}}_\alpha. \tag{15}$$

The feature vector $\mathbf{f}^{\{0\}}(i, j) = \mathbf{s}$ is the set of latent variables describing the $n \times n$ pixel neighborhood centered at site (i, j). In contrast, to use a monolithic set of features, ICA learns a new feature extraction matrix $\mathbf{A}^\dagger$ for each new domain in an unsupervised and totally automated way. Furthermore, the features are independent of one another, resulting in improved estimates of logistic regression parameters $\boldsymbol{\omega}$ during the learning stage.

5. EXPERIMENTAL RESULTS

5.1. A brief summary of the algorithm

Previous sections have provided a very general framework for building an automated object segmentation system. While the general system can be succinctly described by a set of mappings presented in Figure 3, our experiments used the following instantiation of the aforementioned framework. First, the feature extraction matrix $\mathbf{A}^\dagger$ was learned using an unlabeled set of images. Next, given a training image/label pair, the algorithm: (i) extracts features

$\mathbf{f}^{\{0\}}$, using $\mathbf{A}^{\dagger}$, and (ii) produces $\mathbf{L}_{\text{eroded}}$, $\mathbf{L}_{\text{dilated}}$, $\mathbf{L}_{e'}$, $\mathbf{L}_{d'}$ by applying morphological operations on the ground truth image $\mathbf{L}$. Subsequently, five $\mathfrak{L}_0$ classifiers are trained using ICA features as input and label images as targets. The classifiers output probability maps $\mathbf{P}^{\{0\}}_{\text{type}}$, type $\in$ {region, eroded, dilated, e', d'}. A second round of feature extraction is then carried out on the newly extracted probability maps, producing second-order features $\mathbf{f}_{\{0\}}$, that serve as the input to train two $\mathfrak{L}_1$ classifiers. In turn, the second-order classifiers produce two probability maps, $\mathbf{P}^{\{1\}}_{\text{region}}$ and $\mathbf{P}^{\{1\}}_{\text{eroded}}$, used for creating the topological landscape and markers. The last step employs the standard watershed algorithm for producing the final output of the system $\mathbf{L}^{\{\text{ws}\}}$.

5.2. Experimental procedure

To test HS-CDWS, we had a granulometry expert manually label nine, 236 × 637 pixel, images containing oil sand ore (see Figure 1). Using a different set of *unlabeled* oil sand ore images, we learned a generative ICA model using the FastICA algorithm [30]. This ICA model was estimated using 100, 000 randomly selected patches, each 16 × 16 pixels in order to learn 49 Gabor-like filters (resembling those in Figure 4). To provide multiresolution information, two gaussian filters were applied to each ICA filter response, thereby producing 150 features for each pixel (147 multiresolution ICA features + 3 multiresolution raw pixel values from the original image). This constituted $\mathbf{f}^{\{0\}}$, the input to the $\mathfrak{L}_0$ classifiers. The target outputs $\mathbf{L}^{\{0\}}$ included the original ground truth as well as the derived targets depicted in Figure 2. For all experiments a leave-one-out cross validation (LOOCV) testing strategy was used, whereby each system was trained on eight of the nine images with the remaining image used for testing. The procedure was repeated with every image being a test image once.

To reduce computational complexity, for each target output, we trained a set of classifiers, one for each training image. Hence, for each cross-validation fold, we trained $8 \times 5 = 40$ classifiers corresponding to eight training images and five target outputs. This strategy effectively reduced the memory overhead needed for training, since the number of training examples is reduced by a factor of eight. Formally, for test image I_i:

$$\mathbf{P}^{\{0\}}_{\text{type}} = \frac{1}{n-1}\sum_{j\neq i}^{n} h^{\{0\}}_{\text{type},j}, \tag{16}$$

where type $\in$ {region, eroded, dilated, e', d'}. To take advantage of the rich information contained in the probability maps $\mathbf{P}^{\{0\}}$, a second round of feature extraction was carried out, where a bank of gaussian filters was used to extract multiresolution features $\mathbf{f}^{\{1\}}$. To fuse the information into $\mathfrak{L}_1$ probability maps, we trained a set of $\mathfrak{L}_1$ classifiers to produce the mapping: $\mathbf{f}^{\{1\}} \mapsto \mathbf{P}^{\{1\}}_{\text{type}}$, with type $\in$ {region, eroded}. As in [31], we used an internal LOOCV procedure to maximize generalization accuracy. Both $\mathfrak{L}_0$ level and $\mathfrak{L}_1$ level classification were done using logistic regression as implemented by the PrTools [32] Matlab toolbox.

5.3. Evaluation criteria

We used several criteria to evaluate the performance of each algorithm. Respectively, TP, TN, FP, and FN stand for the number of samples (i.e., pixels) being labeled as true positive, true negative, false positive, and false negative.

Intersection-over-union (I/U), for binary labeling A and B, is defined as $|\mathbf{A}\cap\mathbf{B}|/|\mathbf{A}\cup\mathbf{B}| = \mathbf{TP}/(\mathbf{TP}+\mathbf{FP}+\mathbf{FN})$ and is also known as the Jaccard measure.

Pixel accuracy defined as $(\text{TP}+\text{TN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN})$.

Precision defined as TP/(TP + FP) and is also known as positive predictive value.

Recall defined as TP/(TP + FN) and is also known as sensitivity.

Labeling score defined as

$$\mathbf{L} = \min(S(A,B), S(B,A)),$$

$$S(A,B) = \sum_{j}^{m}\left[\sum_{i}^{n}\left(\frac{|A_j\cap B_i|}{|A_j\cup B_i|}\,\frac{B_i}{\bigcup\limits_{|A_j\cap B_i|\neq 0} B_j}\right)\frac{A_j}{\bigcup\limits_{j} A_j}\right], \tag{17}$$

where each A_j is a connected component in image A and each B_i is a connected component in image B. The labeling score is a form of local intersection-over-union, which penalizes errors at both the pixel level *and* at the object level.

5.4. Results

To examine the efficacy of the proposed algorithm, three sets of systems were tested. First, a standard CDWS system (no stacking) was created using ICA features called *ICA-CDWS*. Next, for the *ICA-HS-CDWS* system, we trained $\mathfrak{L}_1$ level classifiers directly on the output of the five $\mathfrak{L}_0$ probability maps produced by classifiers trained on standard ground truth as well as new targets derived from the ground truth. Note that this version of the system did not perform the second round of feature extraction, that is, $\mathbf{f}^{\{1\}} = \mathbf{P}^{\{0\}}$. Finally, the third system, *MR-ICA-HS-CDWS*, had the same setup as the second system, but used the extended set of multiresolution features extracted from $\mathbf{P}^{\{0\}}$. Results, presented in Table 1 and Figure 5, clearly demonstrate the improvement gained by using heterogeneous stacking together with features extracted from $\mathbf{P}^{\{0\}}$. Notice that heterogeneous cascades, with interleaved feature extraction, produce the best results on average and improve upon the scores for essentially every performance metric in every image. The only exception being image 5, where the recall score was slightly degraded by the proposed system. In all other cases, the MR-ICA-HS-CDWS system was able to improve performance in comparison to the base (ICA-CDWS) classification. Interestingly, the recall score for image 5 is one of only two images, where the stacking without feature extraction outperformed stacking with interleaved feature extraction. We believe better features can fix this anomaly and further improve performance. The probability that there are no statistically significant differences in performance as calculated by the

Table 1: Performance comparison of base classification (L0) to heterogeneous stacking (L1). For each experimental condition the tables represent leave-one-out cross-validation results.

(a) ICA-CDWS

Image	jacq	acc	prec	Recall	Label score
1	0.68	0.77	0.79	0.83	0.51
2	0.74	0.83	0.80	0.91	0.62
3	0.73	0.81	0.84	0.84	0.56
4	0.72	0.79	0.86	0.81	0.51
5	0.69	0.78	0.79	0.84	0.52
6	0.76	0.83	0.87	0.86	0.62
7	0.73	0.80	0.84	0.84	0.51
8	0.66	0.76	0.75	0.85	0.54
9	0.73	0.80	0.83	0.85	0.54
Mean	**0.71**	**0.80**	**0.82**	**0.85**	**0.55**
stdev	0.03	0.02	0.04	0.03	0.04

(b) MR-ICA-HS-CDWS

Image	jacq	acc	prec	Recall	Label score
1	0.71	0.80	0.82	0.84	0.59
2	0.77	0.85	0.83	0.91	0.63
3	0.76	0.84	0.87	0.86	0.62
4	0.74	0.81	0.88	0.82	0.54
5	0.71	0.80	0.83	0.83	0.57
6	0.81	0.86	0.89	0.89	0.69
7	0.77	0.84	0.88	0.86	0.53
8	0.71	0.80	0.79	0.87	0.57
9	0.74	0.81	0.85	0.85	0.61
Mean	**0.75**	**0.83**	**0.85**	**0.86**	**0.60**
stdev	0.03	0.02	0.04	0.03	0.05

(c) ICA-HS-CDWS

Image	jacq	acc	prec	Recall	Label score
1	0.70	0.79	0.81	0.83	0.56
2	0.77	0.86	0.83	0.91	0.61
3	0.75	0.83	0.86	0.85	0.61
4	0.74	0.81	0.88	0.82	0.54
5	0.70	0.79	0.81	0.84	0.55
6	0.79	0.85	0.88	0.89	0.65
7	0.75	0.82	0.86	0.86	0.55
8	0.68	0.79	0.77	0.86	0.56
9	0.73	0.81	0.84	0.86	0.56
Mean	**0.74**	**0.82**	**0.84**	**0.86**	**0.58**
stdev	0.03	0.03	0.04	0.03	0.04

students t-test for each performance metric is, respectively: 0.00004, 0.00001, 0.00000, 0.01942, and 0.00049, (for I/U, accuracy, precision, recall, and label scores) indicating that the performance of MR-ICA-HS-CDWS is superior to that of the ICA-CDWS system. In addition, to compare the three aforementioned systems against previous results, Table 2 displays data from the original CDWS research [8]. Several points are immediately apparent. First, the ICA features are weaker than the original hand-crafted features used by CDWS. To some extent this is not surprising, as ICA extracted 49 linear features at three resolutions. In contrast, CDWS utilized 30 hand-crafted nonlinear extraction procedures (e.g., morphological operators) at four resolutions. We believe nonlinear feature extraction methods (e.g., nonlinear PCA) can improve performance and expect to pursue this line of research in the future. However, despite the

Table 2: Performance of OSA, WipFrag, and original CDWS systems against CDWS using ICA and heterogeneous stacking.

System	I/U	Pixel accuracy	Precision	Recall	Label score
OSA	0.68	0.78	0.84	0.79	0.55
WipFrag	0.59	0.65	0.66	0.85	0.36
CDWS	**0.76**	**0.84**	**0.87**	**0.86**	**0.62**
ICA->CDWS	0.71	0.80	0.82	0.85	0.55
HS(ICA)->CDWS	0.74	0.82	0.84	0.86	0.58
MR-HS(ICA)->CDWS	**0.75**	**0.83**	**0.85**	**0.86**	**0.60**

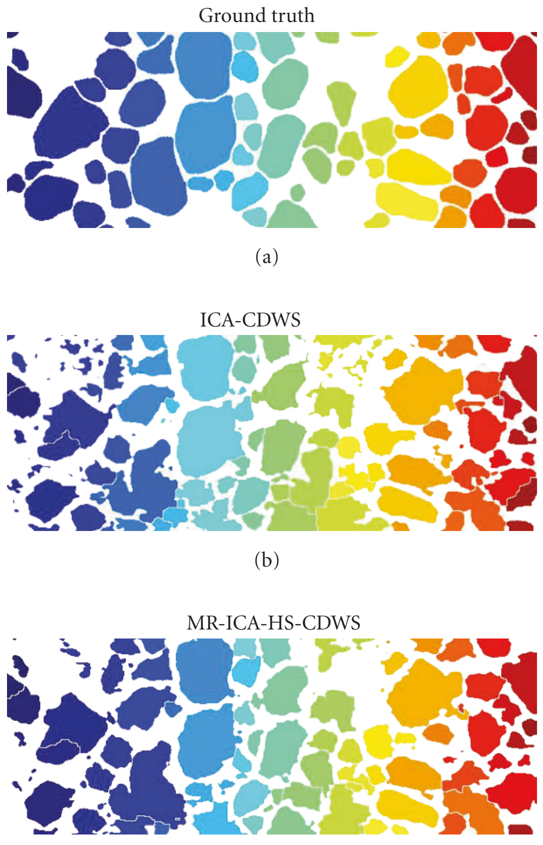

Figure 5: Output for $\mathfrak{L}_0$ and $\mathfrak{L}_1$ layers. Notice the significant reduction in noise as well as the improvement in object-object boundary delineation.

shortcomings of ICA, the MR-ICA-HS-CDWS system, a *fully automated* algorithm, was able to achieve results very similar to those of CDWS utilizing hand-crafted features.

6. CONCLUSION

Our previous paper, [8], proposed a principled machine learning approach, for extracting (i) object markers, (ii) object-background region boundaries, and (iii) topological surface used by the classical watershed algorithm. A major contribution of this paper was to further expose the benefits of manipulating ground truth data by presenting and evaluating heterogeneous stacking. By training a classifiers on transformations of the ground truth—for example, *eroded, dilated, and so on*—the resulting probability maps produced useful components readily utilized by higher-order machine learned classifiers to derive object markers and boundaries. The second contribution of the paper was the application of ICA to automate feature extraction process. By utilizing automated feature extraction in conjunction with heterogeneous stacking, an automated segmentation system can be efficiently constructed with little or no domain knowledge but with performance comparable to state-of-the-art. Furthermore, Section 5 also indicate that additional performance can be achieved by interleaving learning and feature extraction.

ACKNOWLEDGMENT

This research is supported in part by NSERC, Alberta Ingenuity Fund, iCORE, Syncrude Canada Ltd., Matrikon, the Alberta Ingenuity Centre for Machine Learning, and the University of Alberta.

REFERENCES

[1] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," in *Mathematical Morphology in Image Processing*, E. Dougherty, Ed., Marcel Dekker, New York, NY, USA, 1992.
[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2002.
[3] A. Bleau and L. J. Leon, "Watershed-based segmentation and region merging," *Computer Vision and Image Understanding*, vol. 77, no. 3, pp. 317–370, 2000.
[4] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.
[5] J. Fan, G. Zeng, M. Body, and M.-S. Hacid, "Seeded region growing: an extensive and comparative study," *Pattern Recognition Letters*, vol. 26, no. 8, pp. 1139–1156, 2005.
[6] O. Lezoray and H. Cardot, "Bayesian marker extraction for color watershed in segmenting microscopic images," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR '02)*, vol. 1, pp. 739–742, Quebec City, Canada, August 2002.
[7] O. Lezoray and H. Cardot, "Cooperation of color pixel classification schemes and color watershed: a study for microscopic images," *IEEE Transactions on Image Processing*, vol. 11, no. 7, pp. 783–789, 2002.
[8] I. Levner and H. Zhang, "Classification-driven watershed segmentation," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1437–1445, 2007.
[9] D. H. Wolpert, "Stacked generalisation," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer, New York, NY, USA, 2001.
[11] A. Webb, *Statistical Pattern Recognition*, John Wiley & Sons, New York, NY, USA, 2002.
[12] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural Computing*, vol. 3, pp. 79–87, 1991.

[14] Y. Freund and R. E. Schapire, “A decision-theoretical generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[15] Y. Raviv and N. Intrator, “Bootstrapping with noise: an effective regularization technique,” *Connection Science*, vol. 8, no. 3, pp. 355–372, 1996.

[16] L. Breiman, “Randomizing outputs to increase prediction accuracy,” *Machine Learning*, vol. 40, no. 3, pp. 229–242, 2000.

[17] G. Martínez-Muñoz and A. Suárez, “Switching class labels to generate classification ensembles,” *Pattern Recognition*, vol. 38, no. 10, pp. 1483–1494, 2005.

[18] K. M. Ting and I. H. Witten, “Issues in stacked generalization,” *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[19] R. M. Haralick, “Statistical and structural approaches to texture,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.

[20] P. P. Ohanian and R. C. Dubes, “Performance evaluation for four classes of textural features,” *Pattern Recognition*, vol. 25, no. 8, pp. 819–833, 1992.

[21] T. Randen and J. H. Husøy, “Filtering for texture classification: a comparative study,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

[22] R. M. Haralick, K. Shanmugam, and I. Dinstein, “Textural features for image classification,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[23] T. Ojala and M. Pietikäinen, “Unsupervised texture segmentation using feature distributions,” *Pattern Recognition*, vol. 32, no. 3, pp. 477–486, 1999.

[24] F. S. Cohen, Z. Fan, and M. A. Patel, “Classification of rotated and scaled textured images using Gaussian Markov random field models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 192–202, 1991.

[25] J. S. De Bonet and P. A. Viola, “A nonparametric multi-scale statistical model for natural images,” in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10, MIT Press, Cambridge, Mass, USA, 1998.

[26] K. Tieu and P. A. Viola, “Boosting image retrieval,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ’00)*, vol. 1, pp. 228–235, Hilton Head Island, SC, USA, June 2000.

[27] P. O. Hoyer and A. Hyvärinen, “Independent component analysis applied to feature extraction from colour and stereo images,” *Network: Computation in Neural Systems*, vol. 11, no. 3, pp. 191–210, 2000.

[28] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, Wiley-Interscience, New York, NY, USA, 2001.

[29] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural Networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[30] A. Hyvärinen, “Fast and robust fixed-point algorithms for independent component analysis,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[31] P. Paclík, T. C. W. Landgrebe, D. M. J. Tax, and R. P. W. Duin, “On deriving the second-stage training set for trainable combiners,” in *Proceedings of the 6th International Workshop on Multiple Classifier Systems (MCS ’05)*, vol. 3541, pp. 136–146, Seaside, Calif, USA, June 2005.

[32] R. P. W. Duin, P. Juszczak, P. Paclík, E. Pekalska, D. de Ridder, and D. M. J. Tax, “PRTools4, A Matlab Toolbox for Pattern Recognition,” Delft University of Technology, 2004.